Enhancements of the Non-linear Knapsack Cryptosystem

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science

at the
University of Canterbury

by

Zhiqi Tu

University of Canterbury

2006

Examining Committee

Supervisor	Professor Tadao Takaoka Department of Computer Science The University of Canterbury
External Examiner	Professor Clark Thomborson Department of Computer Science The University of Auckland

To my parents and Yi,
for their love and support

Acknowledgements

This project was funded by a Technology for Industry Fellowship (TIF) from the Foundation for Research, Science and Technology through Tait Electronics Ltd., New Zealand. The support is deeply appreciated.

My greatest thanks go to my supervisor, Professor Tadao Takaoka, for his guidance, and tireless help through the duration of my studies.

I would like to thank my parents for their love and support. I would like to also thank Dr. Ian McLoughlin, Dr. Malcolm Shore and Sarah for their big help during my difficult times.

Finally I would like to thank my all friends for their company when I felt tired over the period of my studies. Without all of you, I would not be able to finish this thesis.

Abstract

Nowadays all existing public key cryptosystems are classified into three categories relied on different mathematical foundations. The first one is based on the difficulty of factoring the product of two big prime numbers. The representatives are the RSA and the Rabin cryptosystems. The second one such as the ElGamal cryptosystem is based on the discrete logarithm problem. The last one is based on the NP-completeness of the knapsack problem. The first two categories survived crypto attacks, whereas the last one was broken and there has been no attempt to use such a cryptosystem.

In order to save the last category, Kiriyama proposed a new public key cryptosystem based on the non-linear knapsack problem, which is an NP-complete problem. Due to the non-linear property of the non-linear knapsack problem, this system resists all known attacks to the linear knapsack problem. Based on his work, we extend our research in several ways.

Firstly, we propose an encrypted secret sharing scheme. We improve the security of shares by our method over other existing secret sharing schemes. Simply speaking, in our scheme, it is hard for outsiders to recover a secret even if somehow they could collect all shares, because each share is already encrypted when it is generated. Moreover, our scheme is efficient.

Then we propose a multiple identities authentication scheme, developed on the basis of the non-linear knapsack scheme. It verifies the ownership of an entity's several identities in only one execution of our scheme. More importantly, it protects the privacy of the entities from outsiders. Furthermore, it can be used in resource-constrained devices due to low computational complexity.

We implement the above schemes in the C language under the Linux system. The experimental results show the high efficiency of our schemes, due to low computational complexity of the non-linear knapsack problem, which works as the mathematical foundation of our research.

Contents

List of Figures

Chapter 1

Introduction

1.1 Research Motivation

With the wide use of computers and communication networks such as Internet, more and more commercial activities, traditional businesses or government services, can be implemented on the Internet. However, as we all know, communications over open networks are not secure. An effective solution to secure communication over open networks is to apply cryptographic schemes to protect the transmitted messages. Simply speaking, cryptography is the study of mathematical techniques related to the aspects of information security [41]. Cryptography addresses the following four goals [41]:

1. *Confidentiality*: to protect the content of information from those unauthorized to read it

2. *Data integrity*: to assure no alteration or replacement of data by those unauthorized

3. *Authentication*: to verify the validity of data and the identity of senders

4. *Non-repudiation*: to prevent an identity from denying previous actions

There are two categories of key-based cryptographic algorithms: symmetric algorithms and asymmetric algorithms. The same secret key is used to encrypt and decrypt the information in symmetric algorithms, as Figure 1.1 shows.

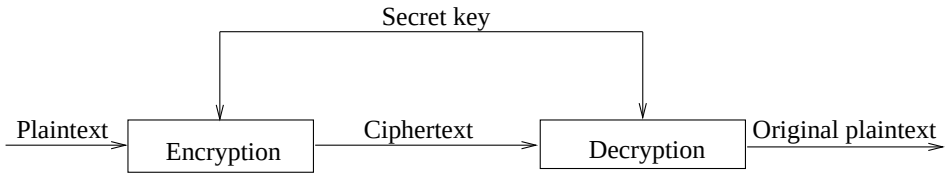

Figure 1.1: Encryption and Decryption with One Secret Key

In asymmetric algorithms, however, two different keys (namely, public key and private key) are used to encrypt and decrypt information, as shown in Figure 2.2.

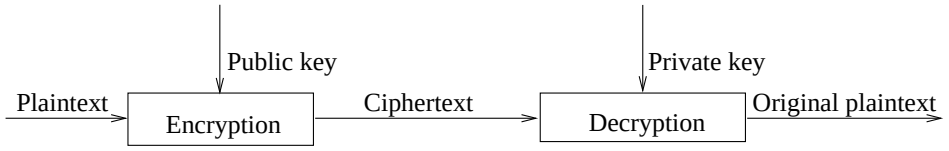

Figure 1.2: Encryption and Decryption with Two Different Keys [54]

As the same (secret) key is used for encryption and decryption in symmetric cryptosystems, it is necessary to transfer the secret key among all communicating parties before secure communication begins. Prior to the birth of asymmetric cryptosystems (also called public-key cryptosystems), the exchange of the secret keys has always been a difficult problem because of the need for a confidential channel. In addition, for each pair of communicating parties, a different pair of the secret keys need to be generated and shared, so the management of the secret keys had also been an issue.

In 1976, Diffie and Hellman [12] were the first researchers in the public area to become aware of public key cryptosystems. In public-key cryptosystems, an important advantage over symmetric cryptosystems is that key exchange can occur without the need for a secure channel [38]. As Diffie and Hellman suggested, public key cryptosystems are based on the one-way trapdoor functions, which means it is significantly easier to

compute in the forward direction than in the inverse direction. However, the inverse direction is easy given a certain piece of trapdoor information [12].

In public-key cryptosystems, an enciphering key (often called public key) can be made public, and a deciphering key (often called private key) is kept secret by the owner. Thus everyone can use the intended receiver's public key to encrypt the message, and send the ciphertext to the receiver. Only the receiver can use his/her own private key to decrypt the ciphertext to obtain the original message. The encryption and decryption processes can be simply denoted as follows:

$$E_{pk}(M) = C$$

$$D_{sk}(C) = D_{sk}(E_{pk}(M)) = M$$

Here M denotes the original message, and C denotes the ciphertext which is sent over the insecure channel. The subscripts pk and sk respectively denote the public key and the private key. The functions E and D respectively represent the encryption and decryption functions. And they satisfy the following properties:

1. Given the information sk and pk respectively to the functions D and E, it is easy to compute the functions D_{sk} and E_{pk}.
2. For each private key sk and its corresponding public key pk, the one way function E_{pk} is the inverse of the function D_{sk}.
3. It is computationally infeasible to compute the function D without the knowledge of sk. Moreover, it is also computationally infeasible to derive any knowledge about sk from the public function E_{pk} and public key pk.

Since the concept of public key cryptosystem was proposed by Diffie and Hellman, there have been several concrete systems. They are classified into three categories according to their reliance on different mathematical problems.

The first one is based on the difficulty of the integer factorization problem. The RSA cryptosystem [52] and the Rabin cryptosystem are two famous representatives in this

category. The former is a most widely used cryptosystem, while the latter is the first cryptosystem whose security has been mathematically proven to be equivalent to the difficulty of factoring. In a later chapter we will discuss further the latter representative.

The second category is based on the difficulty of the discrete logarithm problem. A typical cryptosystem is the ElGamal cryptosystem [13].

The third category is based on the knapsack problem, which is an NP-complete problem. The cryptosystem was proposed by Merkle and Hellman, and was called the Merkle-Hellman Knapsack cryptosystem [42].

The first two categories survived existing crypto attacks, and are said to be safe for practical use. However, the Merkle-Hellman Knapsack Cryptosystem and its many variants were broken by Shamir's algorithm [57], and later by LLL algorithm by Lenstra et. al. [35]. Thus there has been no attempt to use a cryptosystem in the third category to date. However, the Merkle-Hellman Knapsack cryptosystem has the attractive feature that it operates at the high speeds [47]. It can process more than 100 times faster than RSA (with the modulus of about 500 bits), whether hardware or software implementations are used, and thus can rival classical secret key systems in speed [52].

In order to take advantage of its speed, and revive a cryptosystem in the third category that can be used for practical use, Akito Kiriyama [25] recently proposed a new public key cryptosystem based on the non-linear knapsack problem, which is known to be an NP-complete problem. The non-linear property in his scheme can resist existing attacks to the linear knapsack cryptosystems and their variants. Based on his work, we further extend the non-linear knapsack cryptosystem in several ways.

1.2 Research Objectives

By combining the non-linear property of a non-linear knapsack cryptosystem with other mathematical concepts, the following objectives are achieved through our researches:

1. We propose a scheme, called an encrypted (n, n) threshold secret sharing scheme, which combines the concept of the Chinese Remainder Theorem (CRT) and the

non-linear knapsack scheme with the concept of secret sharing. The biggest advantage of our scheme over other secret sharing schemes would be that we increase the security of shares against outsiders. Simply speaking, it would be hard for outsiders to recover the secret even if they could collect n shares, because those shares are encrypted at the same time they are generated.

2. We propose a multiple identities authentication scheme that can be used for access control. Simply speaking, several identities of one entity can be verified in one execution of our scheme. In addition, due to low computational complexity of our scheme, it can be used in resource-constrained devices for access control.

We evaluate the security and performance of each of our schemes. In addition, we implement each one in the C language under the Linux platforms. The experimental results show our schemes are very efficient. More importantly, the non-linear knapsack scheme is far more efficient than the RSA encryption/decryption scheme, due to the low computational complexity of the non-linear knapsack problem.

1.3 Thesis Structure

Chapter 2 in this thesis introduces all related background theories, including the development of cryptography in section 2.1, the mathematical problems of two categories of cryptographic schemes and their typical representatives, which are the integer factorization problem and the Rabin and the RSA Cryptosystems in section 2.2, and the discrete logarithm problem and the ElGamal Cryptosystem in section 2.3, respectively.

In chapter 3, we start with a general overview of the mathematical foundation of the knapsack problem. We cover the theory of NP-completeness in section 3.2, followed by an overview of the Merkle-Hellman Knapsack cryptosystem in section 3.3. Shamir's method for attacking to the Merkle-Hellman Knapsack cryptosystem is briefly explained in section 3.4. The foundation of our research work, Kiriyama's non-linear knapsack cryptosystem, is presented in section 3.5, and the evaluation of his scheme is given in section 3.6. Finally, we summarize the whole chapter in section 3.7.

From chapter 4, we start to present all of our research work. We start with the introduction of an encrypted (n, n) threshold secret sharing scheme. Firstly, the general overview of Shamir's and Blakley's secret sharing schemes are given in section 4.2. Then our scheme is illustrated in section 4.3, followed by the analysis of security of our scheme in section 4.4. Finally, a conclusion is given in section 4.5.

Chapter 5 illustrates our multiple identities authentication scheme used for access control. In section 5.2 related work is discussed, which includes the overview of three ways to implement the access control systems. Then our scheme will be discussed in detail in section 5.3, namely the description of the algorithm and its proof. After that the evaluation is presented in section 5.4, focused on the efficiency and security analysis and comparisons, followed by possible applications of our scheme in section 5.5. A summary is given in section 5.6.

Finally conclusions are presented and future work is discussed in Chapter 6.

Chapter 2

Background

2.1 Introduction of Cryptography

The word Cryptology (stems from Greek roots) means "hidden" and "word", and is an umbrella term used to describe the entire field of secret communications [39]. Cryptology consists of two parts, which are cryptography and cryptanalysis. The part that deals with the design of algorithms, protocols, and systems which are used to protect information against specific threats, is called cryptography [50]. The other part that uses mathematical methods to prove that the design (an implementation of information protection) does not achieve a security goal or that it cannot withstand an attack from the list of threats given in the security specification of the design is called cryptanalysis [50]. Persons who are engaged in research into the corresponding part are respectively called cryptographers and cryptanalysts.

Simply speaking, cryptographers try to find methods to protect information, whereas cryptanalysts try to find attacks to prove that the cryptographers' designs are not secure enough. Cryptographers and cryptanalysts are like players in a game of chess, who compete with each other to become winners. Moreover, cryptographers have to change their roles back and forth from the attackers to the defenders and back. Just like in a game, the sequences of moves and counter-moves need to be considered carefully. Additionally, cryptographers have to consider some situations, in which the opposite

side may break rules or violate expectations. In a word, cryptography is a fascinating discipline because of its game-like adversarial nature [41].

Now let us define several basic terms which are used throughout the whole chapter.

Sender and Receiver: A sender is a transmitter, who wants to send the message to one or more recipients. The intended recipient(s) is(are) called the receiver(s). Only the receiver can read the message from the sender. To an interceptor, the message looks like garbage.

Plaintext and Ciphertext: An original message is called a plaintext, which is readable for anyone. Through some transformations, a plaintext is converted into unreadable form, which is called a ciphertext or cryptogram.

Encryption and Decryption: The process of converting the plaintext into ciphertext is called encryption. And the inverse process, recovering the ciphertext to the original plaintext is called decryption.

Cryptographic algorithm and Key: The step-by-step descriptions of encryption and decryption processes are called cryptographic algorithms (also called ciphers) [41]. Usually, cryptographic algorithms are mathematical functions. And the operations of those functions are controlled by a key, which is a secret piece of information that customizes how the ciphertext is produced.

Cryptosystem: A set of cryptographic primitives such as cryptographic algorithms, keys, and any actions defined by the users and so on implemented together as a system constitutes a cryptosystem [41].

In the next section, we will introduce several earliest forms of ciphers: transposition, substitution ciphers (Caesar cipher) and one-time pad. They are all important contributions to the development of the history of cryptography before 1970s.

2.2 Introduction of Several Ciphers

Cryptography has a long and colorful history, which dates back to thousands of years. The earliest forms of hiding information can be divided into two categories, namely transposition and substitution. Transposition is to rearrange the order of letters in a message. Figure 2.1 [54] shows a simple example of how to use transposition to hide the content of a message.

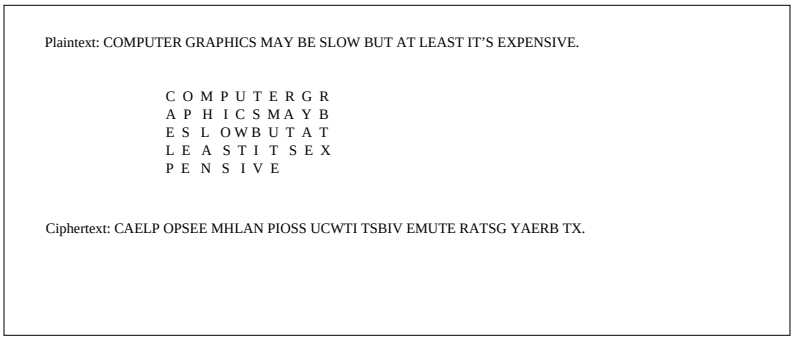

Figure 2.1: A Simple Example of Using Transposition

As we can see from figure 2.1, the plaintext is written horizontally onto a piece of graph paper of fixed width and the ciphertext is read off vertically [54]. To recover the original message, one needs to put the ciphertext vertically onto a piece of paper of exactly identical width, then read the plaintext horizontally.

The second category, substitution means that each letter in a plaintext is substituted for another letter to form the ciphertext [54]. One of the earliest and simplest substitution ciphers is the Caesar cipher used by Julius Caesar, who wrote the letters to Cicero and his other friends. Each letter in the plaintext is replaced by the third (cyclically) later letter in the Latin alphabet [23]. Today, we could express Caesar cipher as the following way [39]:

$$y = x \oplus z$$

where x represents each letter in the plaintext, z represents how far each letter in the plaintext shifts cyclically to later letter, and y represents the letter in the ciphertext. Here $\oplus$ denotes addition modulo 26 (the total number of English characters).

Obviously, the length of the plaintext is the same as that of the ciphertext in those two categories mentioned above. A frequency analysis on the ciphertext would give a good hint to cryptanalysts. Detailed cryptanalysis is discussed in [60, 15].

In 1917, Major Joseph Mauborgne and AT&T's Gilbert Vername [23] invented an unbreakable encryption scheme, which is called a one-time pad. Specifically, the sender generates a large non-repeating set of random letters, which are used as one-time pads. Then encryption is the addition modulo 26 of each letter in the plaintext and each letter in the one-time pads. Note, as the name of the scheme suggests, each letter in the pads is used only once for one message. The following small example (drawn from [54]) illustrates the idea of this scheme:

Here is the plaintext: ONE TIME PAD

And the sequence of the pads is: TBFRGFARFM

Then the ciphertext is: IPKLPSFHGQ, which is produced as follows:

$$Q + T \,(mod\, 26) = I$$

$$N + B \,(mod\, 26) = P$$

$$E + F \,(mod\, 26) = K$$

$$...$$

$$D + M \,(mod\, 26) = Q$$

Since each letter in the pads is randomly generated, it has the same probability for each letter to appear in the pads. That is to say, the cryptanalysts have no way to determine which letter could be a key letter used to encrypt the plaintext. The inventors believe that their method, one-time pad is unbreakable, but they do not give any proofs. In 1949, C. E. Shannon published his paper "Communication Theory of Secrecy Systems [58]", in which he not only proves the unbreakability of one-time pad, but also establishes sharp bounds on the required amount of secret keys that must be transferred securely to the intended recipients when any perfect cipher is used [39].

Shannon's great work did not bring an explosion of research into cryptography. However, this did occur when the paper "New Direction in Cryptography [12]" was published by Diffie and Hellman in 1976. They established the concepts of public key cryptography that continue to be used today [39].

So far, we have introduced several ciphers, which are important to the development of the history of cryptography before 1970s. For those readers who are interested in detail, Kahn's book "The Codebreakers [23]", from the non-technical point of view, gives an interesting introduction. This book traces the history of cryptography from the ancient time (about 4000 years ago) to the twentieth century.

Two important advances were made in 1970s, the publication of Data Encryption Standard (DES) [45], and the development of public key cryptosystems, proposed by Diffie and Hellman. Since then cryptographic algorithms are considered to fall into two categories, namely symmetric cryptographic algorithms, and asymmetric cryptographic algorithms (also called public-key cryptographic algorithms). Section 2.3 presents the first category, including its typical cryptosystems DES and AES [10] (Advanced Encryption Standard, which replaces DES as a new standard in 2001).

After that two typical public key cryptosystems and their mathematical foundations are illustrated in detail in section 2.4, namely, the Rabin public key cryptosystem and ElGamal cryptosystem, together with their respective mathematical foundations, which are the integer factorization problem and the discrete logarithm problem. In section 2.5, we briefly introduce several basic types of cryptanalytic attacks, and then introduce some attacks on the RSA cryptosystem. Finally a conclusion is given in section 2.6.

2.3 Symmetric Cryptosystems

Generally speaking, in symmetric cryptographic algorithms, the same key is used for encryption and decryption. Thus the security of symmetric cryptosystems totally relies on the security of the keys. That is to say, if the keys are revealed to those unintended

recipients, the whole cryptosystem is broken. The encryption and decryption functions can be denoted by:

$$E_k(M) = C$$

$$D_k(C) = D_k(E_k(M)) = M$$

Here M denotes the plaintext, and C denotes the ciphertext. The functions E and D respectively represent the encryption and decryption functions. k denotes the key for the encryption and decryption. Figure 2.2 describes a two-party communication by using the symmetric cryptosystem.

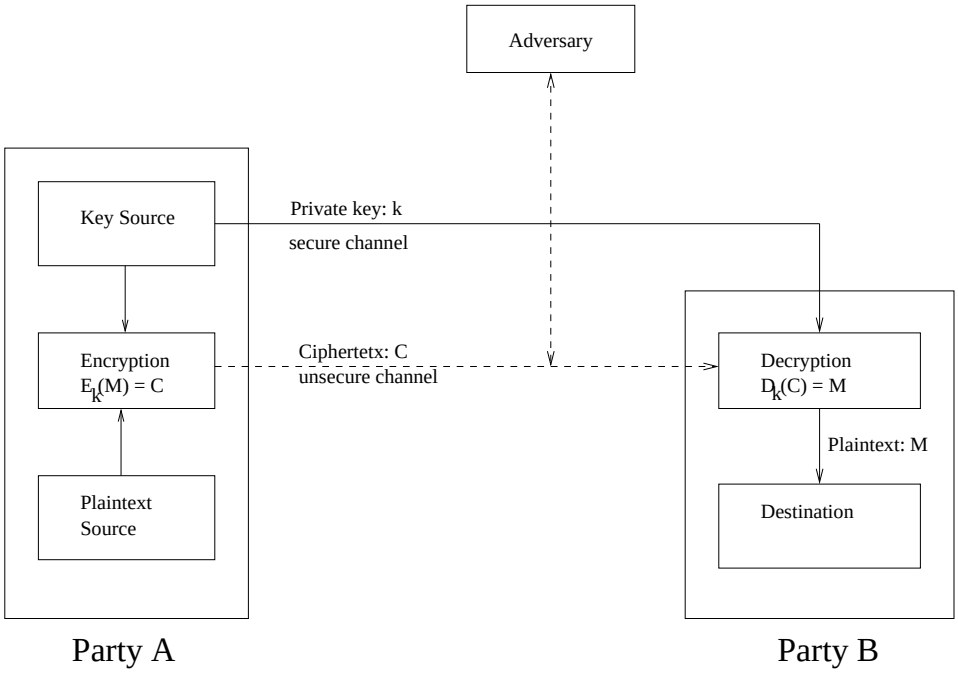

Figure 2.2: Two-Party Communications by Using Symmetric Cryptosystems [41]

As we can see from the above figure, if the key is securely exchanged between two parties (party A as a sender and party B as a receiver) before communication happens, the adversary can not decrypt the ciphertext to obtain the original message.

DES (Data Encryption Standard) is such a typical symmetric algorithm, which has been used as a standard since 1975. Initially the NBS (National Bureau of Standards

in USA), now the NIST (National Institute of Standards and Technology) initiated a program to develop a standard cryptographic algorithm, in order to protect computers and communication data for businesses such as banks and other large financial organizations [54]. IBM submitted a promising algorithm as a candidate. After modifications by the NBS, this algorithm was published in 1975. In the following year this algorithm was adopted as a federal standard (called DES) on November 23, 1976 and authorized for use on all unclassified government communications [6].

Simply speaking, DES is a block cipher, which means the plaintext is grouped into fixed size blocks of bits to be encrypted. In other words, DES encrypts the plaintext in 64-bit blocks to produce the ciphertext in 64-bit blocks. The length of the key in DES is 64 bits, but only 56 bits are used by the algorithm for encryption and decryption, while 8 bits are used for checking parity. The overall structure of DES algorithm is shown in Figure 2.3.

Generally speaking, the plaintext goes in from one end, and then through 16 identical stages of operations (known as rounds), the ciphertext comes out of the other end. Firstly, a 64-bit block of the plaintext is broken into two halves after initial permutation, which are 32 bits for each. Then the keys and the right part are fed into a function F as the input. Through some transformations of F, the result is combined with the left part via an XOR. Thus the new result becomes the right part for the next round. The old right part becomes the new left part. After repeating those operations 16 times, the final left and right parts are joined together to construct a new 64-bit block. Applying the inverse of initial permutation to this block, a block of the ciphertext is produced. The same algorithm and the keys are applied to the decryption to recover the original message.

The actions of the function F are shown in Figure 2.4.

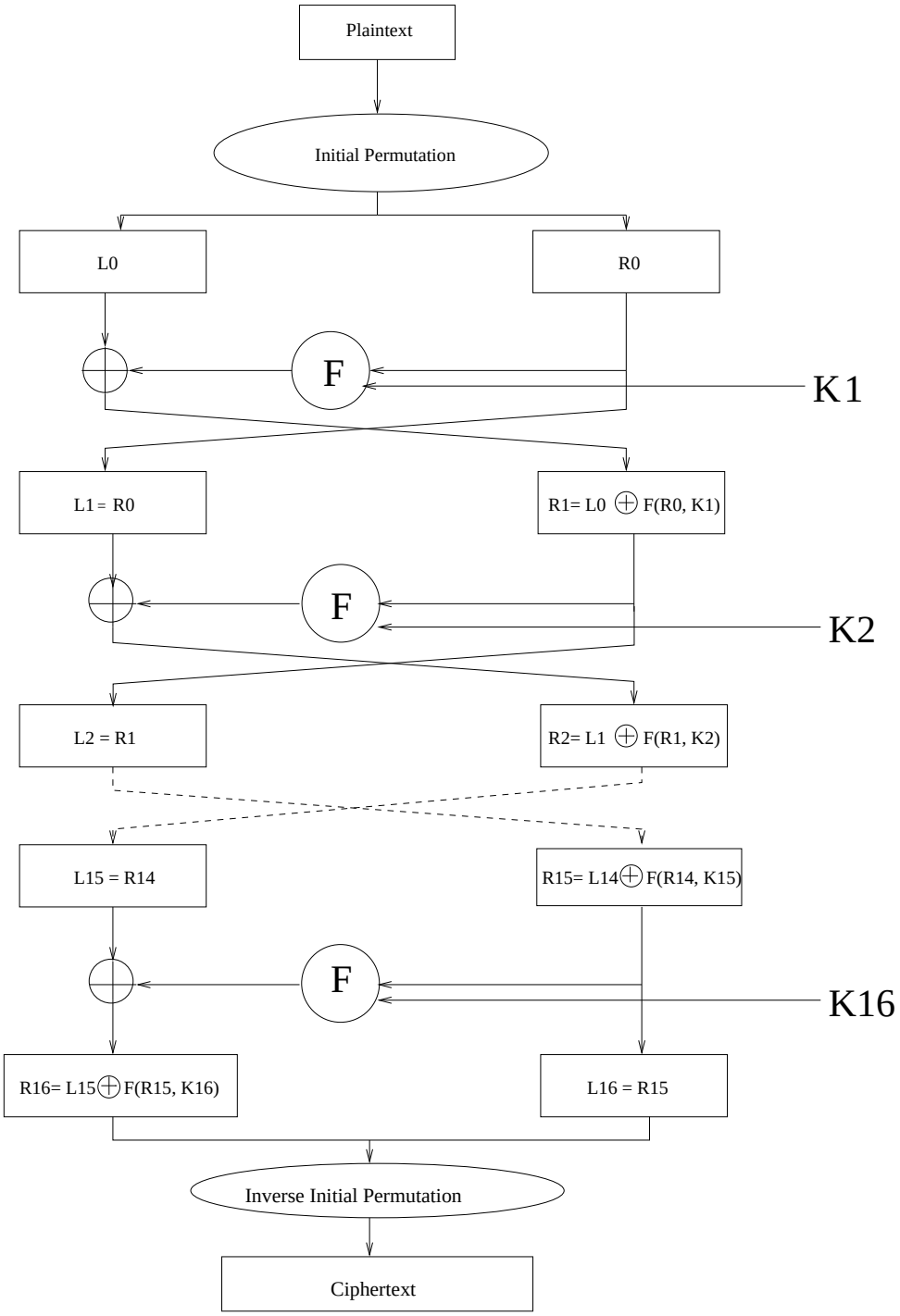

Figure 2.3: The Overall Structure of DES [54]

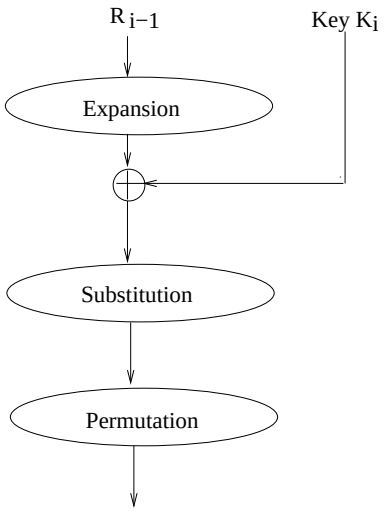

Figure 2.4: The Actions of the Function F [41, 54]

Here the symbols E, S, and P are respectively represented as the operations Expansion, Substitution, and Permutation. The function F can be denoted by

$$F(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i)).$$

Unfortunately, the DES key size has been shown to be insufficient to protect against the brute force attacks, which have been shown to succeed in 56 hours [18]). This means the attackers can test all possible keys through the key space to find the correct one to recover the original plaintext. In other words, the attackers would find the correct key, after on average 2^{63} trials, when the key size is 64 bits. As a result, DES is not secure for use nowadays.

In 2001, AES (Advanced Encryption Standard) was adopted by NIST (National Institute if Standards and Technology) to replace DES as a new standard. AES is a block cipher as well. It has a fixed block size of 128 bits, and its key size could be 128, 192, or even 256 bits. Thus an attack against 128-bit key requires on average 2^{127} trials, which takes unreasonable amount of time. Moreover, AES is fast in both software and hardware, and it is easy to implement. AES as a new encryption standard is currently being adopted on a large scale.

2.4 Public Key Cryptosystems and Their Mathematical Foundations

As we mentioned before, it is necessary in symmetric cryptosystems to transfer secret keys among communicating parties before communication begins. Moreover, keys must be exchanged in a secure way, otherwise, if secret keys are revealed to attackers, the whole cryptosystem would be broken. On the other hand, different pairs of secret keys need to be generated and shared by different pairs of communicating parties. Thus the management of secret keys has been a big issue as well.

An important advantage of public key cryptosystems over symmetric cryptosystems is that key exchange can take place without the need of secure channel [38]. In public key cryptosystems, there are two keys used for each user, a public key and a private key, with the public key for encryption and the private key for decryption. The public key can be sent over an insecure channel. Anyone can use the intended recipient's public key to encrypt the message to produce the ciphertext. Only the intended recipient can use his/her own private key to decrypt the ciphertext to obtain the original message. Anyone else who does not know the private key can not recover the message.

In public key cryptosystems, the private key is always linked to the public key in a mathematical way. Moreover, it has to be computationally infeasible to derive any knowledge about the private key from the public key. Since the concept of public key cryptosystems was proposed by Diffie and Hellman, there have been several public key systems proposed. They are classified into three categories according to their links to the different mathematical problems.

The first one is based on the difficulty of integer factorization. The RSA cryptosystem and the Rabin cryptosystem are two famous representatives in this category. The former is a widely used system. The latter is the first public key cryptosystem whose security has been proven mathematically. Breaking the Rabin cryptosystem is provably equivalent to factoring. We would put emphasis on introducing the Rabin cryptosystem and its mathematical foundation in section 2.4.1.

The second category is based on the difficulty of discrete logarithm. A typical cryp-

tosystem is the ElGamal cryptosystem. In section 2.4.2, the ElGamal cryptosystem and the discrete logarithm problem will be described.

The third category is based on the knapsack problem, which is an NP-complete problem. A cryptosystem was proposed by Merkle and Hellman. In chapter 3, we introduce it in detail.

2.4.1 The Rabin cryptosystem and the integer factorization problem

As previously noted, the RSA cryptosystem is a widely used cryptosystem based on the integer factorization problem. The encryption and decryption methods of the RSA system are as follows:

Randomly generate two prime numbers p and q with roughly the same size, and let the product of p and q be n. Represent the message M as an integer between 0 and $n-1$. Then randomly choose the encryption key e such that e and $(p-1)(q-1)$ are relatively prime. Compute the decryption key d such that $ed \equiv 1 \,(mod\,(p-1)(q-1))$. The numbers e and n are public keys. The numbers d, p and q are private keys.

The processes of encryption and decryption are simple. The sender obtains the public keys and then computes the ciphertext C by $C = M^e \,(mod\, n)$.

The decryption is an inverse process of encryption. The receiver uses his/her private key to recover the original plaintext by $M = C^d \,(mod\, n)$.

The security of the RSA scheme rests on the fact that breaking this system is at least as difficult as factoring the product n. To date, nobody has not found any algorithm to solve the factorization problem in a reasonable amount of time.

Another cryptosystem based on the same mathematical problem has a stronger theoretic foundation. The Rabin cryptosystem is also a public key system, which was proposed in 1979 by Rabin [41]. Like RSA, the Rabin cryptosystem is based on the difficulty of integer factorization. However, the Rabin cryptosystem is the first public

key cryptosystem, for which it has been proven that the ability to decrypt messages is equivalent to the ability to factor large numbers [51]. That is to say, as long as it is impossible in practice to factor large numbers, it will be impossible to decrypt messages [51].

The following illustrates the Rabin cryptographic algorithm with a small example. Then its underlying mathematical problem is given.

parameters:

p, q : Let p and q be two prime numbers with roughly the same size. In order to simplify the computation of square roots modulo p and q (see below), p and q are both chosen to be $p \equiv q \equiv 3 \,(mod\, 4)$. We can also choose $p \equiv q \equiv 1 \,(mod\, 4)$.

n : Let n be the product of prime numbers p and q.

private key:

The prime numbers p and q are both the private key.

public key:

The product n of p and q is the public key.

encryption:

The encryption process is a simple process as follows:

1. The sender represents the message as an integer m in the range $\{0, 1, ..., n-1\}$
2. The sender computes the ciphertext $c = m^2 \,(mod\, n)$
3. The ciphertext c is sent to the receiver

decryption:

The following part illustrates the decryption process:

1. The receiver first computes the square roots of c modulo the primes p and q, which are denoted by
$$m_p = c^{1/2} \, (mod \, p)$$
and
$$m_q = c^{1/2} \, (mod \, q)$$
Since the receiver knows the factorization of n, and $p+1$ and $q+1$ are a multiple of 4, the computations of m_p and m_q are described as follows:
$$m_p = c^{(p+1)/4} \, (mod \, p)$$
and
$$m_q = c^{(q+1)/4} \, (mod \, q)$$
In fact, the first congruence has two possible solutions $\pm m_p$ and so does the second. That is to say, the original message could be constructed by one of four pairs of possible solutions, which are (m_p, m_q), $(m_p, -m_q)$, $(-m_p, m_q)$, and $(-m_p, -m_q)$, respectively. To avoid ambiguity to decide which pair is the correct one to construct the original message, the pre-specified redundancy to the original message such as attaching some bits to the end of the plaintext is always used, prior to encryption.

2. Then the receiver uses the Chinese Remainder Theorem (CRT. Its definition is given in section 4.3) to compute the original message by
$$m = (m_p q q^{-1} + m_q p p^{-1}) \, (mod \, n)$$
$$m = (m_p q q^{-1} + (-m_q) p p^{-1}) \, (mod \, n)$$
$$m = ((-m_p) q q^{-1} + m_q p p^{-1}) \, (mod \, n)$$
$$m = ((-m_p) q q^{-1} + (-m_q) p p^{-1}) \, (mod \, n)$$
With high probability, one message would present the required redundancy. Thus, the receiver can determine the original message.

proof:

Firstly we define several terms, which are used in the following proof. Definitions are drawn from [41].

$\mathbf{Z_n}$: The integers modulo n is denoted by Z_n, which is the set of integers $\{0, 1, 2, ..., n-1\}$.

$\mathbf{Z_n^*}$: The multiplicative group of Z_n is $Z_n^* = \{a \in Z_n\} \,|\, gcd(a, n) = 1\}$. In particular, if n is a prime number, then $Z_n^* = \{a \,|\, 1 \leq a \leq n-1\}$.

quadratic residue modulo n: Let $a \in Z_n^*$, a is said to be a quadratic residue modulo n, if there exists an $x \in Z_n^*$ such that $x^2 \equiv a \,(mod\, n)$. The Legendre symbol $L(a, n)$ is 1 if a is a quadratic residue modulo a prime number n.

The ciphertext c is a quadratic residue of m mod p, and thus Legendre symbol $L(c, p) = c^{(p-1)/2} \,(mod\, p) = 1 \,(mod\, p) = 1$. In addition, m_p and m_q are computed by $m_p = c^{(p+1)/4} \,(mod\, p)$, and $m_q = c^{(q+1)/4} \,(mod\, q)$, since $p+1$ and $q+1$ are a multiple of 4. Then,

$$m_p^2 \equiv c^{(p+1)/2} \equiv c^{(p-1)/2} c \equiv c \,(mod\, p)$$

Note that $-m_p$ also satisfies the above equation. $\pm m_q$ can be computed in the same way.

example:

Let the private keys p and q be 7 and 11, and thus the public key n is 77. The sender takes $m = 11$ as the plaintext. In order to avoid the ambiguity, the sender attaches two bits "01" to the end of the original message. Thus m is 45 as the original message.

The sender computes the ciphertext $c = m^2 \,(mod\, n) = 45^2 \,(mod\, 77) = 23$, and sends c to the receiver.

As the receiver knows the private keys p and q, the square roots m_p and m_q are computed by $m_p = c^{(p+1)/4} \,(mod\, p) = 4$, and $m_q = c^{(q+1)/4} \,(mod\, q) = 1$. Thus four solutions are produced, which are $(4, 1)$, $(-4, 1)$, $(4, -1)$, and $(-4, -1)$ respectively.

Applying the Chinese Remainder Theorem to the above solutions, the receiver obtains four possible values of the original message, namely 67, 32, 45, and 10. Then the receiver represents them as the binary form. Only the value 45 satisfies the required

redundancy. Thus the receiver decrypts the ciphertext c to 45, and then recovers the original message $m = 11$.

underlying mathematical foundation:

As we mentioned before, the security of the Rabin cryptosystem relies on the integer factorization problem. That is to say, the problem faced to the attackers who try to recover the original plaintext from some given ciphertext is computationally equivalent to factoring [41].

Specifically speaking, the integer factorization problem is defined as follows [41]:

> Given a positive integer n, find its prime factorization; that is to write $n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.

For example, an integer 21 can be split into two prime numbers 3 and 7. It is easy to multiply those prime numbers together to get an integer. However, factoring an integer is not an easy job, especially when the multiplier (prime numbers) are big numbers for instance 100 (or more) decimal digits.

Knuth [27] gives an excellent presentation of many factoring algorithms. So far, if a big number is the product of two primes with roughly the same size, there are no known algorithms that can factor this big number in polynomial time. The recent news from the RSA Laboratories reports that a team at the German Federal Agency for Information Technology Security successfully factors a 640-bit number (about 200 decimal digits). This effort takes them approximately 30 2.2GHz-Opteron-CPU years over five months of calendar time [53]. Therefore, factoring a number is regarded as a difficult problem, and the security of several public key cryptosystems such as the Rabin cryptosystem and the RSA cryptosystem both depend on the difficulty of integer factorization.

Compared with other cryptosystems whose security is based on the difficulty of integer factorization, for example the RSA cryptosystem, the Rabin cryptosystem is the first one to be proven that breaking it is as difficult as factoring. The following part gives the proof.

Theorem: If there is a polynomial time algorithm F that can compute the plaintext m from the ciphertext c with some positive probability, we can construct an algorithm that factors the modulo n into two prime factors p and q in polynomially expected time.

Proof: Pick up a random number x in Z_n as the original message. Then compute $y = x^2 \,(mod\, n)$ and input (x, n) into the algorithm F. Then the algorithm F will return $x^* = y^{1/2} \,(mod\, n)$ which is one of four possible values which may be used to construct the original message.

As x is chosen randomly, we will have $prob(x = x^*) = prob(x^* = -x) = 1/4$, which means the probability that the random number x is the same as the plaintext is one-fourth. On average, we can hit the event $x \neq x^*$ or $x \neq -x^*$ at two trials. In this case, we have $gcd(x - x^*, n) = p$ or $gcd(x^* - x, n) = q$. That is to say, we would obtain one of the prime factors of n by computing the value of $gcd(x - x^*, n)$ or $gcd(x^* - x, n)$.

Since gcd is a polynomial time algorithm, we can factor n in a polynomial expected time.

Obviously, the conclusion of the above theorem contradicts the fact that there are no known algorithms that can factor a big number in polynomial time. Thus it in turn proves the security of the Rabin cryptosystem. That is to say, as long as it is impossible in practice to factor a large number in polynomial time, it will be impossible to recover the original plaintext from the ciphertext without knowing the private key.

2.4.2 The ElGamal cryptosystem and the discrete logarithm problem

The ElGamal cryptosystem [13] is a public key system, which was proposed by ElGamal in 1985. Its security is based on the intractability of the discrete logarithm problem. That is to say, if the discrete logarithm problem could be solved efficiently, then the ElGamal system could be broken.

Firstly we will define some related mathematical terms. All definitions are drawn from [41, 54, 50]. The ElGamal system is described, followed by a small example and the proof of this system. Finally the underlying mathematical foundation, which is the discrete logarithm problem, is given.

Group G: A group $G = (S, *)$ is an algebraic structure that satisfies the following conditions ($*$ denotes a binary operation on S):

1. For any two elements $a, b \in S$, $c = a * b \in S$. This property is called *closure*.
2. The group operation is *associative*. That is, $a * (b * c) = (a * b) * c$ for all a, b, c $\in S$
3. There is an element $1 \in S$, called the *identity element*, such that $a * 1 = 1 * a = a$ for $a \in S$
4. For each $a \in S$ there exists an element $a^{-1} \in S$, called the *inverse* of a, such that $a * a^{-1} = a^{-1} * a = 1$
5. For any two $a, b \in S$, $a * b = b * a$, this group is called the commutative group.

Finite Group: A group G is finite if $|G|$ is finite. $|G|$ denotes the number of elements in G.

Order: The number of elements in a finite group G is called its order. The order of a, $(a \in G)$, is defined to be the least positive integer t such that $a^t = 1$, provided that such an integer exists. If such a t does not exist, then the order of a is defined to be ∞.

Subgroup: A non-empty subset H of a group G is a subgroup of G if H is itself a group with respect to the operation of G. If G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$. Hence, if $a \in G$, the order of a divides $|G|$. This fact is called *Lagrange Theorem*.

Cyclic Group: A group G is cyclic if there is an element $\alpha \in G$ such that for each $b \in G$ there is an integer i with $b = \alpha^i$. Such an element α is called a *generator* of G.

A Generator of Z_n^*: Let $\alpha \in Z_n^*$. (Z_n^* is called the multiplicative group of Z_n. The definitions of Z_n and Z_n^* are given in the preceding section.) If the order of α is $\phi(n)$, then α is said to be a generator of Z_n^* (α does not always exist.). $\phi(n)$ denotes the number of integers in the interval $[1, n]$ which are relatively prime to n, for $n \geq 1$. The function ϕ is called the *Euler phi function*.

We can now start to describe the ElGamal cryptosystem using a small example. The proof of the system and its mathematical foundation are given later.

parameters:

p : Let p be a large prime number

g : Let g be a generator of the multiplicative group Z_p^* of integers modulo p. For prime number p, g exists.

x : Let x be a random number, such that $1 \leq x \leq p-2$

private key:

The random number x is a private key

public key:

The prime number p, the generator g, and the value $y = g^x \, (mod \, p)$ are the public keys.

finding a generator g of Z_p^*:

An efficient way to produce the generator g of the multiplicative group Z_p^* of integers modulo p is described as follows:

1. Select a random large prime number q, and then check if $p = 2q + 1$ is a prime number by using Solovay-Strassen probabilistic primality test [3, 61]. Actually, the probability that a randomly selected element $g \in Z_p^*$ is a generator is $\frac{q-1}{2q} \approx \frac{1}{2}$.

***proof*:** From the preceding definitions, we know the order of a multiplicative group Z_p^* is $p-1$, and according to the properties of generators of Z_p^*, the number of generators is $\phi(\phi(p)) = \phi(p-1) = \phi(2q) = \phi(2)\phi(q) = q-1$, if Z_p^* is a cyclic.

In addition the probability of a random element in Z_p^* being a generator is denoted by $\frac{\phi(n)}{n}$, (n denotes the order of a cyclic group.).

Thus, the probability is computed by $\frac{\phi(n)}{n} = \frac{\phi(p-1)}{p-1} \approx \frac{q-1}{2q} \approx \frac{1}{2}$.

2. Select a random element $g \in Z_p^*$ such that $g \neq 1$, then do:

```
for i from 1 to 2
{
    compute b = g^q (mod p);
    if b = 1 then go to the step 2;
    compute b = g^2 (mod p);
    if b = 1 then go to the step 2;
}
return g;
```

encryption:

The encryption process works in the following way:

1. The sender represents the message as an integer m in the range $\{0, 1, ..., p-1\}$.
2. The sender randomly selects an integer k, such that $1 \leq k \leq p-2$.
3. The sender computes a pair of ciphertext $a = g^k \,(mod\, p)$, and $b = m * y^k \,(mod\, p)$.
4. A pair of ciphertext (a, b) is sent to the receiver.

decryption:

The decryption is a simple process as follows:

1. The receiver computes $(a^x)^{-1}\,(mod\,p)$ by using his/her own private key x.
2. Then the receiver recovers the original plaintext m by computing $b*(a^x)^{-1}\,(mod\,p)$.

proof:

Since $b = m * y^k\,(mod\,p)$, and $y = g^x\,(mod\,p)$, b can be expressed as $b = m * (g^x)^k\,(mod\,p)$. In addition, $a = g^k\,(mod\,p)$.

Thus the recovery of the plaintext can be expressed by

$$b * (a^x)^{-1} \equiv (m * (g^x)^k) * (g^{kx})^{-1} \equiv m * g^{xk} * g^{-xk} \equiv m\,(mod\,p)$$

example (drawn from [41]):

Suppose that the generation of the parameters is as follows:

Selects the prime $p = 2357$, and compute a generator $g = 2$ of Z^*_{2357}. Then randomly generate the private key $x = 1751$ and compute the public key by

$$y = g^x\,(mod\,p) = 2^{1751}\,(mod\,2357) = 1185$$

To encrypt a message, the sender represents the message as an integer $m = 2035$, then the sender selects a random integer $k = 1520$ and computes a pair of ciphertext a and b by

$$a = g^k\,(mod\,p) = 2^{1520}\,(mod\,2357) = 1430$$

$$b = y^k m\,(mod\,p) = 1185^{1520} * 2035\,(mod\,2357) = 697$$

A pair of ciphertext $(a, b) = (1430, 697)$ is sent to the receiver.

The receiver computes

$$(a^x)^{-1} * b\,(mod\,p) = (1430^{1751})^{-1} * 697\,(mod\,2357) = 2035$$

Thus, the original plaintext is successfully recovered by the receiver.

underlying mathematical foundation:

Generally speaking, the discrete logarithm is the inverse of the discrete exponentiation in a finite cyclic group [62]. Its definition is given as follows [62]:

Discrete Logarithm: Given a cyclic group G of order n with group operation $\times$ and a generator g, exponentiation in G is defined by [62]:

$$g^x = \overbrace{g \times g \times ... \times g}^{x\, terms}$$

Suppose $y = g^x$, then the discrete logarithm of y to the base g is the integer x, and is denoted

$$x = log_g y$$

Discrete Logarithm Problem: Given a cyclic group G of order n, a generator g in G, and an element $y \in G$, find an integer x such that $g^x = y$ [41].

The integer x is unique because it can only be found modulo the order n in G. For example, let $p = 97$ and a generator $g = 5$. Then Z_{97}^* is a cyclic group of order $n = p - 1 = 96$. Since $5^{32} \equiv 35\,(mod\,97)$, the discrete logarithm of 35 to the base 5 is denoted by $log_5 35 = 32$ in Z_{97}^* .

Modular exponentiation within a group is easy to perform. That is to say, if we obtain the values of the generator g, the integer x and the modulo n, it is easy to compute the value of y by the well-known "square-and-multiply" method. The complexity of bit operations in a cyclic group Z_n^* is $O((lgn)^3)$. Knuth [29] gives the detailed discussion about this method.

However, discrete logarithm is much harder to compute than modular exponentiation. So far, all known methods to compute discrete logarithm in a cyclic group require exponential time. Therefore, the intractability of the discrete logarithm problem is regarded as the foundation for the security of a class of cryptosystems for example the ElGamal cryptosystem. For those readers who are interested in the discrete logarithm problem, please see the references [40, 46, 48].

2.4.3 The integer factorization problem and the discrete logarithm problem

Many algorithms for computing discrete logarithms are analogous to integer factorization algorithms. LaMacchia and Odlyzko [34] notes that if the modulo is a prime

number, then the complexity of finding discrete logarithm in cyclic group G is essentially the same as factoring an integer n of about the same size, where n is the product of two approximately equal-length primes. In other words, suppose that n is a composite integer, if the discrete logarithm problem in Z_n^* can be solved in polynomial time, then n can be factored in expected polynomial time [41].

So far, the security of the cryptosystems such as the Rabin cryptosystem and the ElGamal cryptosystem are respectively based on the intractability of the integer factorization problem and the discrete logarithm problem. It is widely believed that both mathematical problems are hard to solve in polynomial time, because nobody has found efficient algorithms to solve them in expected polynomial time so far. However, in the future, if unexpected new mathematical algorithms are found, they would potentially solve both the integer factorization problem and the discrete logarithm problem [49], which in turn destroy the security of those cryptosystems based on those two mathematical problems.

2.5 Cryptanalysis

As we mentioned in section 2.1, cryptology consists of two parts: cryptography and cryptanalysis. The former is to deal with the design of algorithms, protocols, and systems which are used to protect information against specific threats. The latter is to use mathematical methods to prove that the design does not achieve a security goal or that it cannot withstand an attack from the list of threats given in the security specification of the design.

The general definitions for each of several basic types of cryptanalytic attacks are given as follows:

1. Ciphertext-only attack. As it name indicates, the cryptanalyst only has the ciphertexts of several messages, all of which have been encrypted using the same encryption algorithm [54]. By observing the ciphertexts, the cryptanalyst tries to recover the messages as many as possible, or even to deduce the key used to

encrypt the messages. In the history of cryptography, statistical techniques such as frequency analysis were developed for the ciphertext-only attack. Early cipher implemented using pen-and-paper could be broken by this method [11].

2. Known-plaintext attack. The cryptanalyst not only has the ciphertexts, but also has the corresponding plaintext to those ciphertexts. If the cryptanalyst is able to deduce the key used to encrypt the plaintexts, the attack would be successful. Classical ciphers are typically vulnerable to known-plaintext attack [11]. As we mentioned in section 2.2, a Caesar cipher could be broken by a frequency analysis on the ciphertext and corresponding plaintext, and exhaustive search for the small key space (there are only 26 keys/English characters in the Caesar cipher).

3. Chosen-plaintext attack. A chosen-plaintext attack is a model in which cryptanalyst has capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts [64]. This attack model becomes quite important in the context of public key cryptography, because the encryption keys are public. Attackers can encrypt any plaintexts that they want and then obtain the ciphertexts. Generally speaking, any cipher that can prevent chosen-plaintext attacks is then also guaranteed to be secure against known-plaintext and ciphertext-only attacks [64].

4. Chosen-ciphertext attack. A chosen-ciphertetx attack is primarily applicable to asymmetric cryptographic algorithms [54]. The cryptanalyst starts with the ciphertexts to be decrypted with unknown keys, and then obtains the corresponding decrypted plaintexts. Subsequently, the cryptanalyst tries to deduce more information about the key.

5. Related-key attack. This is an attack model that the cryptanalyst observes some ciphertexts encrypted under several keys. Then the mathematic relationship connecting the keys might be known to the attackers [11]. For example, the cryptanalyst may know the last bit of the keys is always the same, even though the cryptanalyst still does not know the exact value of the keys.

The security of a cryptographic algorithm depends on how successful these types of

attacks are. Take the cryptographic scheme RSA for example; since its initial publication, it has survived successfully over 20 years although its security has been analyzed by many researchers. The most considered for RSA is the factoring of the public key. So far it has not been proven that breaking the RSA algorithm is equivalent to factoring a large number. However, it has not been proven either that it is not equivalent. Moreover, no polynomial-time algorithm for factoring large integers on a classical computer has yet been found. The recent report from the RSA Lab reports that factoring 640-bit number takes about 30 2.2GHz-Opteron-CPU years over five months of calendar time [53].

There exists another possible attack by guessing the value $(p-1)(q-1)$. However, as the author in [64] points out that this attack is no easier than factoring n. Also in [1], it shows that recovering even certain bits of information from an RSA-encrypted ciphertext is as hard as decrypting the entire message [54].

On the other hand, there exist some attacks against the implementations of RSA. One of the early weaknesses found was in a system where the users in a group share the same public modulus $n(= pq)$. However, the encryption key e_i and the decryption key d_i are different for each of the users. That is to say, the encrypted messages intended for the user Alice can only be decrypted by Alice, because no one else except for Alice knows the corresponding decryption key. This scheme seems to work well. However, as Simmon shows in [19] this is incorrect and insecure, because another user Bob can use his own exponent e and d to factor the modulus n. Once n is factored successfully, the whole system is broken. Therefore, Simmon points out that an RSA modulus should never be used by more than one entity.

Another attack on RSA's implementation is caused by the use of small exponent encryption key e. For the improvement of efficiency of encryption, it is desirable to select a small encryption exponent e such as $e = 3$ [41]. Let us imagine the following scenario. A sender Bob wants to send the encrypted message M to k ($k \geq e = 3$) persons. Each person has the same encryption exponent $e(= 3)$ but different modulus n_i. Suppose that the message M is less than all modulus n_i. Bob encrypts the message M by using each person's public keys (e, n_i) and sends ciphertexts $C_i = M^3 \, (mod \, n_i)$ to each

person. If an attacker could collect at least three ciphertexts C_i where

$$C_1 = M^3 \,(mod\, n_1)$$

$$C_2 = M^3 \,(mod\, n_2)$$

$$C_3 = M^3 \,(mod\, n_3)$$

since these modulus are most likely pairwise relatively prime, by the Chinese Remainder Theorem, it must be the case satisfying $C' = M^3 \,(mod\, n_1 n_2 n_3)(C' \in Z_{n_1 n_2 n_3})$. As we assumed above that the plaintext message is less than all n_i, we have $M < n_1 n_2 n_3$. Thus the equation $C' = M^3$ holds [5]. Hence, the attacker can recover the plaintext by computing the integer cube root of C'. The cryptosystem is broken when a small encryption exponent is used.

As we mentioned in the beginning of this section, no successful attacks on RSA have been found over 20 years, although its security has been analyzed by many cryptanalysts. The above attacks are only against the implementation of RSA, not against the algorithm itself. Therefore, currently RSA is widely used and is believed to be secure given sufficiently large modulus.

2.6 Conclusion

Cryptography has a long history, which traces back to thousands of years. The earliest forms of ciphers are discussed in section 2.2, namely transposition cipher, substitution cipher(Caesar cipher) and one-time pad.

With the publication of DES and the development of public key cryptosystem in 1970s, cryptosystems became divided into two types of key-based algorithms, namely symmetric cryptographic algorithms and public key cryptographic algorithms. For the former, the same key is used for encryption and decryption. It is discussed in section 2.3.

As for the latter, two different keys (private key and public key) are used. The public key is used for encryption, which can be published openly. The private key is used

for decryption, which has to be kept secret, known only to the owner of the key. Thus, key distribution and key management are not significant issues for the public key cryptosystems.

Generally speaking, public key cryptosystems can be classified into three categories based on different mathematical foundations. One is based on the difficulty of integer factorization. The typical systems are the Rabin cryptosystem and the RSA cryptosystem. Because the Rabin cryptosystem is the first public key system whose security has been mathematically proven to be equivalent to the difficulty of factoring, we put more emphasis on introducing it and its mathematical foundation in section 2.4.1.

The cryptosystem of the second category is based on the discrete logarithm problem. The typical cryptosystem in this category is the ElGamal system. We introduce this system and the discrete logarithm problem in section 2.4.2. After that, in section 2.4.3, we discuss the future of the above problems.

Definitions of basic types of attacks are briefly mentioned in section 2.5. Also several specific attacks on the RSA cryptosystem are given in this section.

In the next chapter, we are going to describe the third category of cryptosystems and its mathematical foundation in detail. A possible attack on the system is also mentioned.

Chapter 3

A Non-linear Knapsack Public Key Scheme

3.1 Introduction

In the preceding chapter, we discussed two categories of public key cryptosystems. The first category is based on the difficulty of integer factorization. The typical cryptosystems of the first category are the RSA system [52] and the Rabin system [51]. The second category is based on the difficulty of discrete logarithm. A typical cryptosystem in this category is the ElGamal system [13]. In this chapter, we discuss the third category of public key cryptosystem, which is based on the knapsack problem, an NP-complete problem. The cryptosystem was proposed by Merkle and Hellman, and was called the Merkle-Hellman Knapsack cryptosystem [42]. Then we are going to introduce a new public key cryptosystem, called *a Non-linear Knapsack Public-Key Cryptosystem* [25, 26], which is the foundation of our work.

An attractive feature of the Merkle-Hellman Knapsack system is its high efficiency. The RSA system is very slow by comparisons [47]. When n is 100, a value recommended in Merkle and Hellman's original paper, the system is more than 100 times faster than RSA system (with the modulus of about 500 bits) in both hardware and software. Its speed can even compete against traditional symmetric cryptosystems.

Unfortunately, the Merkle-Hellman Knapsack system and its many variants were broken by Shamir's algorithm [57] and later by LLL algorithm [35], whereas the other two categories survived existing crypto attacks, and are said to be safe for practical use [26]. Due to the failure, researchers seemed not to focus their efforts on knapsack cryptosystems any further. Moreover, in 2000, IEEE adopted P1363: Standard Specifications for Public-Key Cryptography [21], the discussions about the knapsack cryptosystems were notably missing. This demonstrates the cryptographic community's opinion on and the reality about knapsack cryptosystems: they are no longer important [33].

In order to revive a cryptosystem in the third category, Kiriyama [25] recently proposed a new public-key cryptosystem based on NP-completeness of non-linear knapsack problem, and called it the Non-linear Knapsack Cryptosystem. We believe this new scheme brings theoretical breakthroughs to the research of knapsack cryptosystems. We extend his work in three ways, which will be discussed in the following chapters.

In this chapter, we start with a general overview of the mathematical foundation of knapsack problems: *the theory of NP-completeness* in section 3.2, followed by an overview of Merkle-Hellman Knapsack Scheme in section 3.3. Then Shamir's method for attacking to the Merkle-Hellman Knapsack Scheme is briefly explained in section 3.4. After that, Kiriyama's non-linear knapsack scheme is presented in section 3.5, and the evaluations of his scheme, which is mainly focused on security and efficiency analysis, are given in section 3.6. Finally, we summarize the whole chapter in section 3.7.

3.2 The Theory of NP-completeness

Before we introduce the theory of NP-completeness, let us start with an interesting example, drawn from a famous book, *Computers and Intractability: a Guide to the theory of NP-Completeness* [16].

Suppose that one day, your boss asks you to design an efficient algorithm to a problem. Several weeks later, you have not been able to come up with any algorithms substantially better than brute force searching through all possible solutions, which takes years of computation time. Certainly you do not want to tell your boss:

Sorry, I can not find an efficient algorithm. I guess I am just too dumb.

In order to avoid being fired by your boss, you'd better to say:

I can not find an efficient algorithm, because no such algorithm is possible!

However, unfortunately, proving the above announcement is as hard as finding an efficient algorithm. Another good way is to announce to your boss:

I can not find an efficient algorithm, but neither can all these famous people.

That is to say, you prove to your boss that your problem is as hard as many other problems, which are widely recognized as being difficult. Even famous experts can not find an efficient algorithm for it. Thus, at least you let your boss understand even hiring an expert does not make any differences.

The theory of NP-completeness is such a technique for proving that a given problem is "*just as hard as*" a large number of other problems that are widely recognized as being difficult and that have been confounding the experts for years [16].

Firstly, we are going to introduce the class P in section 3.2.1, followed by the introduction of the class NP and NP-completeness in section 3.2.2. After that, the relationships of P, NP, and NP-completeness are given in section 3.2.3.

3.2.1 The class P

The problems in P can be solved in polynomial time. That is to say, given the input size n, the worst-case running time for them is $O(n^k)$ for some constant k. For example, sorting problems can be done in $O(nlogn)$ when n numbers need to be sorted.

Without loss of generality, the class P is defined on decision problems that are solvable by polynomial time deterministic algorithms. By this we mean that the answer in polynomial time to decision problems is simply "yes" or "no". Take a sorting problem for example, its decision version might be : "Given a sequence of numbers, is this sequence in increasing order? " We think of problems that are solvable by polynomial time deterministic algorithms as being tractable, or easy [9], where $O(n)$ and $O(nlogn)$ are classified in the same polynomial time complexity.

The definition of the class P provides a useful hint to us. If a decision problem is not in the class P, it must be intractable, or hard. That is to say, this problem can not be solved in polynomial time. The class NP contains such problems. We will talk about the class NP in detail in the next subsection.

Polynomials have nice "closure" properties [30]. If an algorithm for a complex problem consists of several algorithms for the simple problems, then the complexity of that algorithm is always bounded by the sum or the product of the complexities of its several component algorithms, since polynomials are closed under addition, multiplication, or composition.

There exists another property for the problems in the class P. If a problem can be solved in polynomial time in one model, it can be solved in polynomial time in another [9]. For example, if a class of problems can be solved in polynomial time on random access machines, they are also solvable in polynomial time on Turing machines, which are basic symbol-manipulating devices (for the detailed descriptions, please see [37, 20]).

3.2.2 The class NP and NP-completeness

As mentioned above, there are some problems that can not be solved in polynomial time. For example, the Travelling Salesman Problem (TSP) belongs to this class. TSP can be simply stated as follows:

> *A salesman wishes to make a tour among a set of cities, n, visiting each city exactly once and finishing at the city he starts from. There is an integer*

cost $c(i, j)$ to travel from the city i to the city j, and the salesman wishes to make the tour whose total cost is bounded by B. Does there exist a tour of all the cities having cost B or less?

For n cities TSP, there are $(n-1)!$ different tours. As picking any cities as the first, there are $(n-1)$ choices for the second city visits, $(n-2)$ choices for the third, and so on. Thus, the number of solutions becomes extremely large for large n. So far people can not find any solutions to solve this problem in polynomial time. By this we do not mean that such algorithms do not exist. What we mean is there are no known polynomial time algorithms, that have been found by scientists so far.

However, if someone claims to have found an instance for the above TSP for which the answer is *"yes"*, it is easy for us to verify the truth or falsity of his claim merely by checking whether the total cost of his tour is bounded by B or not. The computational time of our verification process is polynomial. In other words, we do not care how much time is spent in searching one possible tour among all tours. We just verify in polynomial time whether the answer for the problem is *"yes"*.

Therefore, informally speaking, the algorithms for the problems in the class NP can be divided into two stages. One is a *guessing* stage; the other is a *checking* stage. That is to say, we may guess all possible solutions in the first stage, for the given problem. Then taking each of them as the input to the second stage, we check the truth or falsity of each of them in polynomial time by verifying whether the answer to the problem is "yes" or "no". We call such a class of problems for which there are polynomial bounded nondeterministic algorithms to be the class NP.

NP-complete problems, generally speaking, are the hardest class of problems that are in the class NP. In addition, if there exists a polynomial algorithm for any one NP-complete problem, then there would be a polynomial algorithm for each problem in NP. In theory, any known NP-complete problem can be used to prove a new problem is NP-complete, by using the method of reductions, or transformations. For example, suppose that we want to solve a problem A_1 and we already have an algorithm for the problem A_2. Suppose that we also have a function T that takes an input x for A_1 and produces $T(x)$ in polynomial time, an input for A_2 such that the answer for A_1 on x

is "yes" if and only if the answer for A_2 on $T(x)$ is "yes". Then by combining T and the algorithm for A_2 we have an algorithm for A_1. Figure 3.1 shows the reduction of problem A_1 to problem A_2. A_2 is as hard as A_1. In other words, if A_2 is NP-hard, so is A_1.

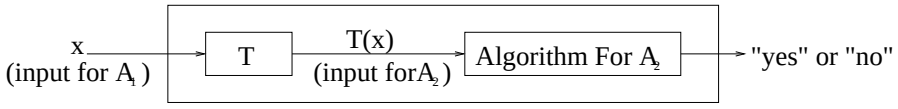

Figure 3.1: Reductions of Problem A_1 to A_2 [16]

The first NP-complete problem is provided by Cook's theorem, which is called the SATISFIABILITY (SAT) problem. It is described as follows:

> Is there a truth assignment, i.e. a way to assign the values *true* or *false*, for the variables in a Boolean expression so that the whole expression has value *true* [30]?

After the first NP-complete problem is proposed, a series of problems have been shown to be NP-complete, through reductions or transformations. The following figure 3.2 shows the transformations of some of them. Noticeably, the knapsack problem is referred to as an NP-complete problem, which provides the mathematical foundation for the Merkle-Hellman knapsack cryptosystem. We will describe it in detail later.

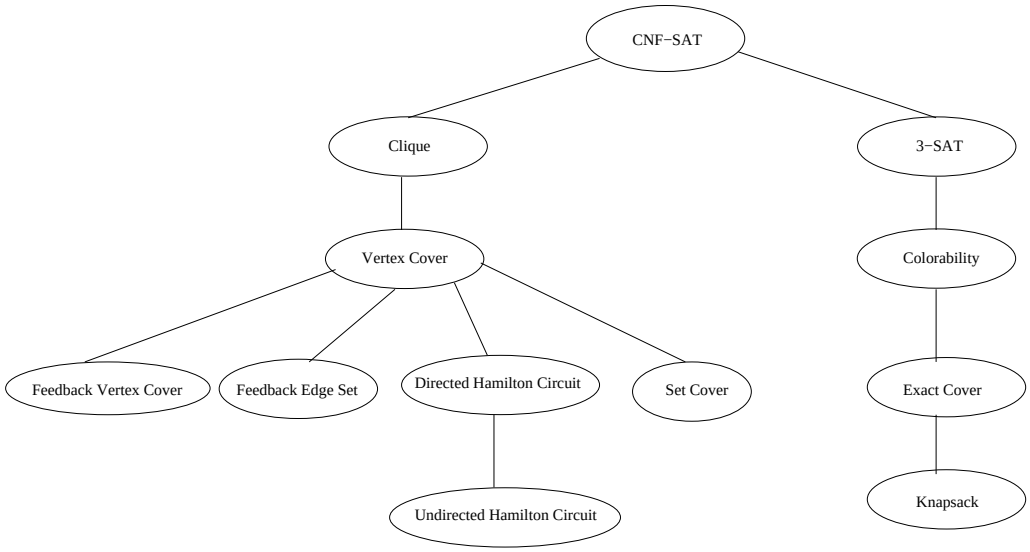

Figure 3.2: Transformations of NP-complete Problems [28]

3.2.3 The relationships among the class P, NP, and NP-complete

The first relationship between P and NP is P $\subseteq$ NP. As we can see from the above discussions, any polynomial time nondeterministic algorithm in the class NP consists of two stages, namely the guessing stage, and the checking stage. Any polynomial time deterministic algorithm in the class P can be used as the checking stage. In other words, a deterministic algorithm A can be regarded as the second stage of a nondeterministic algorithm without the guessing stage. For example, if A is a deterministic algorithm for a problem in the class P and runs in polynomial time, we can obtain a polynomial time nondeterministic algorithm for the problem, just taking A as the checking stage and ignoring the guessing stage. Thus we can claim that A is a polynomial time nondeterministic algorithm for the problem, which implies the problem belongs to the class NP as well.

It is not surprising that a big open question, "Does P = NP or P $\neq$ NP hold? " arises. So far, researchers have not found any proof to the above question. However, it is extensively believed that the equation P $\neq$ NP holds, because NP seems to be more powerful than P, and researchers have not found any methods to convert NP to P.

If there was a polynomial time algorithm for an NP-complete problem, then there would be a polynomial time algorithm for each problem in NP [30]. That is to say, the equation P = NP will hold.

Based on the above discussions, the relationships among P, NP, and NP-completeness are shown as follows, when P is not equal to NP:

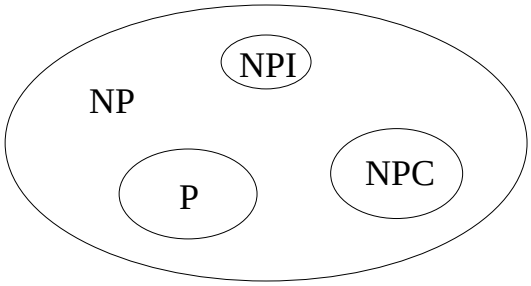

Figure 3.3: The World of NP Problems, unless P = NP [9]

NPC and P respectively denote the class NP-complete and the class P. Both are wholly obtained within the class NP and P $\cap$ NPC = $\emptyset$ [9]. NPI is a class of problems that has not yet been proved either to belong to P or NP-complete [16].

3.3 The Merkle-Hellman Knapsack Cryptosystem

In this section, we will give a general overview about the knapsack problem in section 3.3.1, followed by a detailed description of the Merkle-Hellman knapsack cryptosystem.

3.3.1 The knapsack problem

The knapsack problem can be simply stated as: Given a set of integers $A = \{a_1, a_2, ..., a_n\}$ and an integer S, we need to decide whether there is a subset of A that sums to exactly S [63]. That is defined as the following formula:

$$\sum_{i=1}^{n} x_i a_i = S, \ (x_i \in \{0,1\} \ for \ all \ i)$$

In the worst case, it is a very difficult problem to solve (also called as the Hard-Knapsack problem), requiring an exponential amount of time [63]. However, if a set of integers A is a super-increasing sequence, which has the property that each a_i is larger than the sum of the preceding $i-1$ elements, this problem can be solved in $O(n^2)$ time (called as the Easy-Knapsack problem), by using the following algorithm (called the decryption algorithm for our convenience):

$i = n$;

while $(i \geq 1)$ do the following:

{

If $S \geq a_i$ then $x_i = 1$, and $S = S - a_i$;

Else $x_i = 0$;

$i - -$**;**

}

Return $\{x_1, x_2, ...x_n\}$

We examine each a_i, for $(i = n, n-1, ..., 1)$. If $a_i > S$, we cannot choose a_i. Otherwise we choose a_i and decrease S by a_i. The process proceeds with S decreased as necessary until we finish. Note that each a_i is given by n bits, thus it takes $O(n)$ to subtract a_i from S. Therefore, it takes $O(n^2)$ time for the examination of n numbers.

Based on the NP-completeness of the knapsack problem, Merkle-Hellman proposed a public key cryptosystem. Anyone who tries to solve the Hard-Knapsack problem requires an exponential amount of time, whereas those who solve the Easy-Knapsack problem only require $O(n^2)$, which is polynomial time.

3.3.2 The Merkle-Hellman knapsack cryptosystem

The main idea of the Merkle-Hellman knapsack cryptosystem is to transform an Easy-Knapsack problem to a Hard-Knapsack problem, while keeping the transformation parameters secret.

Specifically, the user generates a super-increasing sequence $\{a_i\}$ as his/her private key. Then he/she generates a multiplier w and a prime number p such that $p > \sum_{i=1}^{n} a_i$, and $gcd(w, p) = 1$. The multiplier w and prime number p are the secret parameters. The sequence $\{a_i\}$ is converted to $\{a'_i\}$ by $a'_i = a_i * w \,(mod\, p)$, for $i = 1, ..., n$. Thus, the sequence $\{a'_i\}$ is no longer super-increasing, but random-looking. It is published as the user's public key.

When someone wishes to send the information to this user, he transforms the plaintext to the vector $\mathbf{x}(= x_1x_2...x_n)$, which is a binary vector of the length n. Then he computes $C = \sum_{i=1}^{n} \mathbf{x} * a'_i$ and sends C to the receiver. The intended receiver computes $M = C * w^{-1} \,(mod\, p)$ first (w^{-1} is the multiplicative inverse of w mod p), and then applies the above decryption algorithm to recover the vector $\mathbf{x}$, which can be recovered into the original plaintext.

Initially, when this scheme was established in 1978, Merkle and Hellman claimed that it was secure. Anyone who does not know the multiplier w, the prime number p, and the multiplicative inverse w^{-1} of w mod p, has great difficulty in decrypting the vector $\mathbf{x}$, based on the difficulty of solving an NP-complete knapsack problem. Unfortunately this scheme was broken later, due to the linearity of the problem, and the property of super-increasing sequence. The next section presents Shamir's method for attacking to the above scheme.

3.4 Shamir's Attacks to the Merkle-Hellman Knapsack Scheme

The security of the Merkle-Hellman knapsack scheme lies in hiding information by using the public key, so that only the user who knows the private key can recover the ciphertext in polynomial time. Otherwise it is an NP-complete problem.

However, Shamir broke this scheme without the knowledge of original super-increasing sequence, a multiplier w, and a prime number p. His attack method is based on the fact that any public key that is obtained from a super-increasing sequence has infinitely many super-increasing private keys that can decipher all messages [11].

That is to say, it is possible to find another pair of numbers, W and P to replace the original multiplier w and the prime number p (also called modulus). Then through the public key sequence $\{a_i'\}$, the attacker computes another super-increasing sequence $\{a_i''\}$ (to replace the original super-increasing sequence $\{a_i\}$) so that this new sequence can be used to decrypt the ciphertext. The following small example (from [11]) illustrates how Shamir's attack method works.

Supposed that we have a super-increasing sequence $\{171, 196, 457, 1191, 2410\}$ as a public key, a multiplier w of 2550, and a modulus p of 8443. Thus we obtain the enciphering key $\{5457, 1663, 216, 6013, 7439\}$.

Given that the vector $\mathbf{x} = (01011)$, the sender computes $C = 15115$. The intended receiver obtains $M = C * w^{-1} \, (mod \, p)$, that is $M = 15115 * 3950 \, (mod \, 8443) = 3797$, $(w^{-1} = 3950)$. Then the receiver applies the decryption algorithm, which is easy for him to recover the vector $\mathbf{x} = (01011)$.

Now if we multiply each element a_i' in the enciphering key by $46 \, mod \, 77$, which is another pair of numbers, a multiplier and a modulus, we obtain a sequence $\{2, 37, 3, 14, 6\}$. After ordering, this is a super-increasing sequence $\{2, 3, 6, 14, 37\}$, which can be used to decrypt the ciphertext.

We compute $M = 15115 * 46 \, (mod \, 77) = 57$. Then we apply the decryption algorithm to M. We can compute a vector $\mathbf{x} = (00111)$. After we re-order the vector sequence

to the original order, the vector $\mathbf{x}$ actually is (01011), which is the same result as that of the intended receiver.

As we can see from this example, there exists at least one pair of numbers, a multiplier W and a modulus P, different from the original one, that can be used to generate another private key from the public key in the public place. Shamir's main contribution was to notice that the Lenstra's integer programming theorem [36] could be used to find at least one pair of parameters W and P in polynomial time. Once these numbers are found, the attackers can decrypt the ciphertext easily. For those people who are interested on the detailed attack method, please see [47, 57, 11]. Another attack method is called LLL algorithm [35] proposed by Lenstra, et. al., which is more widely accepted for the breaking of such knapsack cryptosystems and their variants.

3.5 A Non-linear Knapsack Scheme

3.5.1 A non-linear knapsack scheme

To save the knapsack cryptosystems, Kiriyama proposed in March 2005 a non-linear knapsack scheme [25], which overcomes the weaknesses of the linear knapsack cryptosystem. The following illustrates the non-linear knapsack problem, and then demonstrates a non-linear knapsack scheme with a small example.

Let $f_i(x)$ be a non-linear function of x, and $\mathbf{x}$ be an m-ary vector $\mathbf{x} = (x_1 x_2 ... x_n)$. Each x_i is regarded as a symbol of the alphabet $\{1, 2, ..., m\}$. The non-linear knapsack problem is presented as follows:

$$M = \sum_{i=1}^{n} f_i(x_i)$$

That is to say, given an integer M, the solution for the above equation is to choose appropriate kinds of item $1, 2, ..., n$ to make M, whereas in linear knapsack problem mentioned before, the solution is to choose an item or not to make M. The vector $\mathbf{x}$ is a binary vector of length n in the latter.

The following demonstrates the non-linear knapsack scheme with a small example.

parameters:

n : number of items

m : number of kinds

l : number of mask bits for each item

$mask(i)$: mask pattern of item i, has ln bits of which l bits are 1 and the rest are 0. $mask(i)$ has the following properties:

$$mask(i)\&mask(j) = (00...0),\ all\ 0\ for\ i \neq j$$

$$\&\ denotes\ binary\ bitwise\ and\ operation$$

$$Bitwise\ union\ of\ all\ mask(i)\ is\ (11...1),\ all\ 1$$

$f_i(j)$: mapping from j to $value(i, j)$ is denoted by $f_i(j)$. Let $value(i, j)$ be a vector not greater than $mask(i)$ as a binary vector, where order $\mathbf{x} \leqq \mathbf{y}$ for $\mathbf{x} = x_1...x_n$ and $\mathbf{y} = y_1...y_n$ is defined by $x_i \leqq y_i$ for $i = 1, ..., n$.

p : a prime number such that $p > 2^{ln}$

w : a multiplier, such that $gcd(w, p) = 1$

w^{-1} : a multiplicative inverse of w mod p

lemma:

$mask(i)\&f_i(j) = f_i(j)$

$mask(i)\&f_k(j) = (00...0)$, all 0 for $k \neq i$

private key:

$$f = \begin{pmatrix} (f_1(1), & f_1(2), & ..., & f_1(m)) \\ (f_2(1), & f_2(2), & ..., & f_2(m)) \\ (..., & ..., & ..., & ...) \\ (f_n(1), & f_n(2), & ..., & f_n(m)) \end{pmatrix}$$

For convenience, we call $f_i(j), ((i = 1, ..., n),\ (j = 1, ..., m))$ as a kind.

public key:

$$f' = \begin{pmatrix} (f'_1(1), & f'_1(2), & ..., & f'_1(m)) \\ (f'_2(1), & f'_2(2), & ..., & f'_2(m)) \\ (..., & ..., & ..., & ...) \\ (f'_n(1), & f'_n(2), & ..., & f'_n(m)) \end{pmatrix}$$

Let f' be obtained by operating "w times mod p " on each kind in f. Each kind (denoted by $f'_i(j), ((i = 1, ..., n),\ (j = 1, ..., m))$) in f' can be computed by $f'_i(j) = f_i(j) * w\,(mod\,p)$, and thus each kind in f can be computed by $f_i(j) = f'_i(j) * w^{-1}\,(mod\,p)$.

encryption:

Let the vector $\mathbf{x} = (x_1 x_2 ... x_n)$ be an m-ary sequence of length n to be encrypted. The ciphertext C is computed by

$$C = \sum_{i=1}^{n} f'_i(x_i)$$

decryption:

The intended receiver computes $M = C * w^{-1}(mod\,p)$ first. Then he/she computes $y_i = f_i^{-1}(mask(i)\&M)$ for $i = 1, ..., n$, and let $\mathbf{y} = (y_1 y_2 ... y_n)$ be the decrypted message.

example from [25]:

$n = 4$, $m = 3$, and $l = 2$. Four mask patterns $mask(i)$, $(i = 1, 2, 3, 4)$ are randomly generated as follows:

$mask(1) = (01001000)$

$mask(2) = (10010000)$

$mask(3) = (00100001)$

$mask(4) = (00000110)$

Based on the above mask patterns, $n * m$ kinds in the private key f are generated.

$$f = \begin{pmatrix} (00001000, & 01001000, & 01000000), \\ (10010000, & 10000000, & 00010000), \\ (00000001, & 00100000, & 00100001), \\ (00000100, & 00000110, & 00000010) \end{pmatrix} = \begin{pmatrix} (8, & 72, & 64), \\ (144, & 128, & 16), \\ (1, & 32, & 33), \\ (4, & 6, & 2) \end{pmatrix}$$

Then the kinds in the public key f' are computed by $f' = f * w(mod\, p)$. Let p be 283, and w be 200, and thus w^{-1}mod p is 75

$$f' = \begin{pmatrix} (185, & 250, & 65), \\ (217, & 130, & 87), \\ (200, & 174, & 91), \\ (234, & 68, & 117) \end{pmatrix}$$

If we set the vector $\mathbf{x} = (1231)$, then the ciphertext C is computed by $C = \sum_{i=1}^{n} f_i'(x_i)$ $= f_1'(1) + f_2'(2) + f_3'(3) + f_4'(1) = 185 + 130 + 91 + 234 = 640$.

The ciphertext C is sent to the receiver. The receiver computes $M = C * w^{-1}(mod\, p) = 640 * 75\, mod\, 283 = 173$. The binary representation of decimal number 173 is 10101101. Then the receiver applies the "bitwise AND" operation to each mask pattern $mask(i)$ and M. The result is shown as follows:

10101101&$mask(1)$ = 10101101&01001000 = 00001000→kind 1

10101101&$mask(2)$ = 10101101&10010000 = 10000000→kind 2

10101101&$mask(3)$ = 10101101&00100001 = 00100001→kind 3

10101101&$mask(4)$ = 10101101&00000110 = 00000100→kind 1

Therefore, the decrypted vector $\mathbf{y} = (1231)$, which is identical to the original vector $\mathbf{x}$. The decryption process is successful.

3.5.2 Proof

As we mentioned before, the original message is transformed into the vector $\mathbf{x}$ to be encrypted, and the vector $\mathbf{y}$ is recovered to construct the decrypted message. Thus, we have to prove the vector $\mathbf{y}$ to be identical to the vector $\mathbf{x}$.

theorem:

$\mathbf{y} = \mathbf{x}$

proof:

$y_i = f_i^{-1}(mask(i)\&M)$

$= f_i^{-1}(mask(i)\&(C * w^{-1}\,(mod\,p)))$

$= f_i^{-1}(mask(i)\&(\sum_{j=1}^{n} f_j'(x_j) * w^{-1}(mod\,p)))$

$= f_i^{-1}(mask(i)\&\sum_{j=1}^{n} f_j(x_j))$

$= f_i^{-1}(mask(i)\&\bigcup_{j=1}^{n} f_j(x_j))$

$= f_i^{-1}(f_i(x_i))$ from lemma

$= x_i$

In the above "$\bigcup$" is similar to "$\sum$" and bitwise "or" operations are done instead of addition.

At first glance, the mapping from each x_i in the vector $\mathbf{x}$ to y_i is not onto. Thus it could not guarantee f_i^{-1} exists for each element i. However, in the Kiriyama scheme, each x_i is a chosen integer; although this choice is not as easy as expected. By this we mean, we choose each kind in the item without repetition. When keys are set up, one-to-one corresponding from x_i to y_i exists. For each x_i, this scheme could guarantee that there is only one y_i in the corresponding range such that $y_i = f_i(x_i)$. Therefore, each decrypted message would correspond to only one plaintext.

3.6 Evaluation

In this section, the security considerations are presented in section 3.6.1, followed by efficiency analysis in section 3.6.2.

3.6.1 Security considerations

Firstly, we have to avoid the trivial cases (including all 0) by using just a few of 2^l bit patterns in $mask(i)$ for the kinds of item i. Secondly, we should also avoid the case that kinds are disjoint from each other in each mask pattern $mask(i)$. The reason for that is we might introduce a super-increasing property on some kinds. Thus it would become vulnerable to the super-increasing attack, as Shamir did to the Merkle-Hellman Knapsack cryptosystem.

On the other hand, if we take heavily overlapping to each kind in the mask pattern $mask(i)$, it would invite an equal-sum event attack. Specifically, when we produce each kind in the private key, if we have l bits for each mask pattern $mask(i)$, then we scatter no more than l ones to the mask locations in the private key, and make the rest of the bits 0. We define the mask locations as the places that bits are 1 in each mask pattern $mask(i)$. For example if $mask(i)$ is (01001000), then mask locations are the second and fifth bits (counting from left to right). We scatter no more than $l(= 2)$ ones to the second or/and fifth bits, and make the rest of bits be 0. Thus kinds (01000000), (00001000), and (01001000) could be three possible kinds in the private key.

Suppose we have four different kinds in the private key (a, b, c, d) such that the following equation holds: $a+b = c+d$. Let (a', b', c', d') be the corresponding kinds in the public key. It is evident that the equation $a' + b' - (c' + d') = kp$ holds for some k. If an attacker can find another set of public key's kinds satisfying the above "equal-sum event", they can easily compute the value of p by using the Euclidean algorithm, which causes the breaking of the whole system. Therefore, in order to avoid this case, we suggest scattering $l/2$ ones to each kind in the private key. Moreover, the number of kinds m could not be large, because the number of cases "equal-sum event" is given by the formula: $\sum_{i=0}^{m} \sum_{j=0}^{m-i} C(m, i)C(m - i, j) = 3^m$. That is to say, among m kinds in each item, the probability of that case is the sum of the number of combinations of i out of m kinds plus that of j out of the rest of kinds, $m - i$. It is obvious that the probability that two l-bits sequences are equal is 2^{-l}. Therefore, the above situation is bounded by $(3^m 2^{-l})^2$, which is equal to $(2^{1.58m-l})^2$. For $m = 10$ and $l = 20$, this probability is about $1/300$. We further multiply this probability by n, the number of

items. Then the probability is about 1/4 [26]. We have an "equal-sum event" at every four settings. Thus, after the private key is set up, we have to exhaustively check the case of "equal-sum event" for security.

The LLL method does not apply to this non-linear knapsack scheme, because this scheme is based on the non-linear property, not the linear property.

Additionally, it is worth to note that Kiriyama has not shown how to reduce an arbitrary instance of some NP-complete problems to an instance of the non-linear knapsack problem. In other words, an unknown plaintext attack on Kiriyama's cryptosystem is not NP-hard until a suitable transformation is supplied. Therefore, Kiriyama's system might not be secure although no one has proven such a reduction is not possible yet. Actually the widely used cryptosystem RSA shares the fate. No one has mathematically proven the security of RSA wholly depends on the problem of factoring. However, nobody proves it does not. In the future, if somebody proposes a method to prove the reduction of NP-complete problem to the non-linear problem is possible, that would be a theoretical breakthrough not only for the Kiriyama scheme itself, but also for the mathematic field.

Another missing element in Kiriyama's scheme is an average-case hard instance-generator. So far, we know of no algorithm which will efficiently generate difficult instances for any NP-complete problems. In any event, for practical cryptosystems, key pairs must be based on problem instances which are hard with a high degree of certainty. As noted in Chapter 2 of this thesis, Rabin has shown that suitably-hard instances are readily constructed, complete with trapdoors, for the integer factorization problem. However, a suitable instance generator is not yet available for Kiriyama's scheme.

The final thing we want to mention here is that Chor-Rivest cryptosystem [8] is the only knapsack-like cryptosystem that has not been broken so far, when its parameters are chosen carefully [47]. Up to date, even Quantum machine can not break NP-complete problems in P time. Thus, any cryptosystem related to NP-complete problems such as Chor-Rivest cryptosystem and non-linear knapsack cryptosystem should have more promising future. Furthermore, these cryptosystems might still be secure in the near future unless new attacking approaches are proposed.

3.6.2 Efficiency analysis

As we mentioned at the start of this chapter, the Merkle-Hellman knapsack scheme is very efficient, 100 times faster than the RSA scheme. Kiriyama's non-linear knapsack scheme keeps advantage of efficiency.

If we use the standard $O(n^2)$ time for the multiple-precision multiplication and division of n-bit integers, the non-linear knapsack scheme only takes $O(n^2)$ for encrypting/decrypting the given message. However, the RSA scheme requires $O(n^3)$ time for encrypting/decrypting a message of the same size.

Some experiments were carried out to compare the running time of encryption and decryption between the non-linear knapsack scheme and the RSA scheme. We set the size of modulus in the RSA scheme to be about 100 decimal digits, which is thought to be safe for the scheme itself. As for the non-linear knapsack scheme, we set the number of items, n, to be 16, the number of kinds, m, to be 10 and the number of mask bits, l, to be 20. Thus the size of modulus in the non-linear knapsack scheme is about 100 digits, which has a comparable size to that of the RSA scheme.

The following table shows the result of our experiments, when we encrypt/decrypt a file with 2KB. All experiments are running on the Linux machine with a Celeron process with 2.2 GHz. The implementation of both schemes is in the C language.

	RSA		Non_linear	
	E(Encryption)	D(Decryption)	E(Encryption)	D(Decryption)
1st Experiment (Sec.)	5.314	5.541	0.021	0.062
2nd Experiment (Sec.)	5.558	5.710	0.021	0.063
3rd Experiment (Sec.)	4.928	4.992	0.023	0.066
Aver. Time (Sec.)	5.267	5.414	0.022	0.064
E+D Aver. Time (Sec.)	10.681		0.086	

Table 3.1: Comparisons About the Running Time

As can be seen from the above table, the total time of encryption and decryption in the non-linear knapsack scheme is about 125 times faster than that in the RSA scheme.

The memory size for the non-linear knapsack scheme is about $lmn^2 = 6KB$, which is reasonable.

3.7 Conclusion

In this chapter, we started with a general overview of the theory of NP-completeness. Then the Merkle-Hellman knapsack scheme was presented, followed by Shamir's attack against the Merkle-Hellman knapsack scheme. After that, the non-linear knapsack scheme and its correctness proof for decryption are illustrated in detail. Finally we evaluate the non-linear knapsack scheme from two aspects: security consideration, and efficiency analysis.

Chapter 4

An Encrypted (n,n) Threshold Secret Sharing Scheme

4.1 Introduction

The motivation for secret sharing schemes originates from the problems associated with exchanging secret keys. Secret sharing schemes enhance the reliability of safeguarding keys or other secrets without increasing risk. The following example will illustrate the concept of secret sharing [65]. Suppose you and your friend accidentally discover a fortune, but neither of you are ready to carry it at that moment. So you two decide to draw a map and go home for preparation. Then when you are ready, you both return to carry the fortune home. Now the question is who is going to keep the map. Apparently, both you and your friend do not really trust each other, and you are afraid that if the other keeps the map, he/she will go alone and take the fortune. Therefore, both of you need to find a way to keep the map so that you can each ensure the other one will not go back alone. It is to split a map into two pieces and make sure the location can be specified only when two pieces are provided. Then you and your friend get one each and happily go home to do your preparations.

Formally speaking, the idea of secret sharing is to start with a secret, and divide it into pieces called *shares* which are distributed amongst users such that the pooled shares

of specific subsets of users allow reconstruction of the original secret [41].

In a secret sharing scheme, there is one *dealer* and n *participants*. The *dealer* is a person who sets up the scheme and distributes the secrets to the *participants*. The *participants* hold their own shares until t (for threshold, $t \leq n$) of them or more decide to pool their shares together in order to recover the secret. The recovery of the secret is accomplished by a person/device, so-called a *combiner*, who collects t or more shares to compute the secret. However, the collections of $t-1$ or fewer shares by a *combiner* do not allow the secret be recovered. Such a secret sharing scheme is called (t, n) threshold secret sharing scheme. The security of this scheme is based on an assumption that it is impossible or hard for an attacker to get t or more shares, which causes the exposure of the secret.

In 1979, Shamir [56] and Blakley [4] independently devised threshold schemes. They both addressed the above issue of safeguarding the cryptographic keys, although their approaches to solving this issue were different. Since the notion was proposed by them, many other secret sharing schemes have been presented, for instance, Asmuth and Bloom [2], Brickell [7] and Karin et. al [24]. The majority of them are (t, n) threshold schemes.

We propose an (n, n) threshold secret sharing scheme by combining the concept of Chinese Remaindering Theorem (CRT, defined in section 4.3) and the non-linear knapsack scheme with the concept of secret sharing. The main advantage of our scheme is to increase the difficulty of recovering the plaintext for attackers outside of the group, because each share is encrypted already by using the non-linear knapsack scheme which was presented in chapter 3. That is to say, even if attackers obtain all n shares somehow, it is still hard for them to recover the plaintext. However, a weakness of our scheme is that we cannot prevent attacks from the inside of the group. In other words, if a group member collects n shares, then he/she can easily recover the plaintext.

We start with the general overviews of Shamir's and Blakley's secret sharing schemes in section 4.2. Then our scheme is illustrated in section 4.3, followed by the analysis of security in section 4.4. Finally, a conclusion is given in section 4.5.

4.2 General Overviews of Shamir's and Blakley's Secret Sharing Schemes

As we mentioned before, Shamir and Blakley first proposed the notion of secret sharing schemes independently in 1979. Their approaches were different, but they both addressed the ways of solving the issue of safeguarding cryptographic keys. Simply speaking, Shamir used Lagrange polynomial interpolation to design (t, n) threshold secret sharing scheme, whereas Blakley used projective spaces to construct a (t, n) threshold scheme. We present both schemes in more details in this section.

4.2.1 Shamir's (t, n) threshold secret sharing scheme

For simplicity, let us start with the design of a $(2, n)$ secret sharing scheme. Suppose we want to share a secret S among n participants, and only when any 2 or more participants pool their shares together, the secret S can be recovered by a combiner. Figure 4.1 shows the idea using the coordinate system.

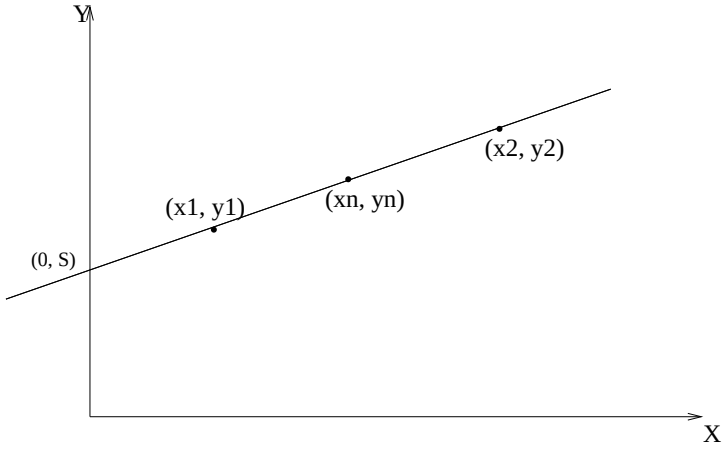

Figure 4.1: Shamir's Scheme

Firstly, a dealer selects the point $(0, S)$ on the Y axis, which corresponds to the secret S. Then he/she randomly draws a line through this point, and picks any n points

$(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ on this line. Each point represents a share. The dealer securely distributes those n shares to n participants. As we all know, two points uniquely determine a line. Therefore, when any two participants pool their shares together, that line is uniquely determined, so that the secret is recovered successfully. One participant on his/her own cannot know that line and the secret. Similarly, a $(3, n)$ threshold secret sharing scheme is designed as follows: a curve through the secret S can be determined by any 3 points, not 2 points or 1 point. Therefore, when any 3 or more participants pool their shares together, the secret can be disclosed.

Formally speaking, for (t, n) threshold secret sharing scheme, we use the curve that corresponds to a $(t-1)$ degree polynomial:

$$f(x) = a_{t-1} * x^{t-1} + a_{t-2} * x^{t-2} + ... + a_1 * x + a_0$$

This curve goes through the Y axis at the point, $(0, S)$, which represents the secret S. Then the dealer randomly chooses n points and securely distributes them. Any t points uniquely determine that curve. The Lagrange interpolation formula allows us to determine the polynomial $f(x)$ of degree $(t-1)$ from the t different points $(x_{i_j}, f(x_{i_j})) = (x_{i_j}, S_{i_j})$ for $j = 1, ..., t$, thus

$$f(x) = \sum_{j=1}^{t} S_{i_j} \prod_{1 \le l \le t, l \ne j} \frac{x - x_{i_l}}{x_{i_j} - x_{i_l}}$$

Noticeably, the polynomial is defined over a finite field Z_p, which is formed by the polynomial $f(x)$ modulo a prime number p. Therefore, the computation of the polynomial $f(x)$ can be expressed by

$$f(x) = \sum_{j=1}^{t} S_{i_j} \prod_{1 \le l \le t, l \ne j} (x - x_{i_l}) * (x_{i_j} - x_{i_l})^{-1} \, (mod\, p)$$

The secret $S = f(0)$, therefore we obtain

$$S = a_0 = \sum_{j=1}^{t} S_{i_j} b_j,$$

where $b_j = \prod_{1 \le l \le t, l \ne j} x_{i_l} * (x_{i_l} - x_{i_j})^{-1} \, (mod\, p)$ [50] (suppose p is a prime number).

Any attackers who collect $t-1$ shares cannot compute the secret S because the $t-1$ equations cannot determine t unknowns. Therefore, Shamir's scheme is a perfect secret sharing scheme (the definition of "perfect scheme" will be given later in section 4.3.) because the knowledge of $t-1$ shares does not provide any advantage to an opponent over knowing no pieces [41]. In addition, it is also an ideal secret sharing scheme, which means the size of shares is equal to that of the secret.

4.2.2 Blakley's (t, n) threshold secret sharing scheme

As we mentioned before, Blakley also devised a secret sharing scheme by using a different approach. His scheme used projective spaces to construct a (t, n) threshold secret sharing scheme [50].

As we all know, two nonparallel lines in the same plane intersect at exactly one point. Three nonparallel planes in space intersect at exactly one point. More generally, any n n-dimensional hyperplanes intersect at a specific point. Therefore, a dealer hides a secret in any single coordinate of the point of intersection. Then each participant is given enough information to define a hyperplane. When t or more of them decide to recover the secret, a combiner computes the plane's intersection point. Any $t-1$ or fewer shares can not determine the secret. However, $t-1$ shares provide some information to narrow the secret down to the line because they intersect to a line. Figure 4.2 presents the idea of Blakley's secret sharing scheme. L_1, L_2, and L_3 are three lines in 3 different hyperplanes, which intersect at a point representing a secret S.

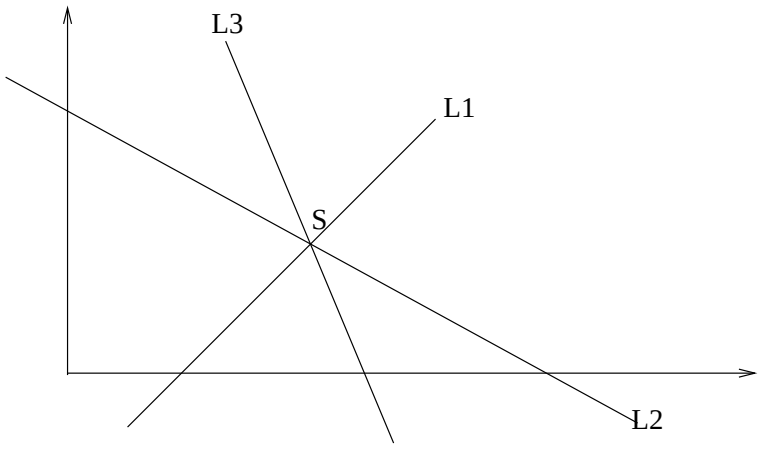

Figure 4.2: Blakley's Scheme

Compared with Shamir's scheme, Blakley's scheme is not perfect because $t-1$ shares do provide some information to an opponent. Thus the more participants collude, the more risk there is that the points will be exposed. However, this scheme can be modified to achieve perfect security by adding restrictions [59].

4.3 Our Scheme: (Encrypted) (n, n) Threshold Secret Sharing Scheme

Before we start to describe our scheme, we mention the Chinese Remainder Theorem [41].

> Chinese Remainder Theorem (CRT): Let n_1, n_2, ..., n_r be pairwise co-prime, where $n = n_1 * n_2 * ... * n_r$. Then the system of congruences
>
> $$x \equiv x_i \,(mod\, n_i),\ i = 1, ..., r$$
>
> has a unique solution x in $\{0, ..., n-1\}$

That is to say, the CRT asserts the equivalence of the representation of integers in modular arithmetic, i.e. x mod n is equivalent to the vector representations $(x_1, ..., x_r)$

[28]. The solution x to the simultaneous congruences in the CRT may be computed as $x = \sum_{i=1}^{r} x_i N_i M_i (mod\, n)$, where $N_i = \frac{n}{n_i}$, and $M_i = N_i^{-1} (mod n_i)$.

Let us illustrate it with a small example with $r = 2$. Let p and q be two prime numbers, and let N be the product of p and q. Suppose M (denoted as a positive integer) is the message to be shared, such that $M < N$. Let $x = M\, mod\, p$, and $y = M\, mod\, q$. Then M is uniquely determined by the formula: $M = xqq^{-1} + ypp^{-1} (mod\, N)$, where p^{-1} is the multiplicative inverse of p mod q, and q^{-1} is the multiplicative inverse of q mod p.

Combining the concept of CRT with the non-linear knapsack scheme, we develop an encrypted secret sharing scheme. That is to say, each share generated by a dealer is encrypted using a non-linear knapsack scheme, before it is distributed to all participants. Thus, even if all shares are collected by an attacker outside of the group, the secret is still hard to recover.

The following illustrates our scheme. For simplicity, we go with a $(2, 2)$ secret sharing scheme first, and we omit the definition of the parameters that appeared in the non-linear knapsack scheme.

parameters:

n_1, n_2 : let them be two large prime numbers

N : let N be the product of n_1 and n_2, and $gcd(w, N) = 1$, where w is a random number as a multiplier

n_1^{-1}, n_2^{-1} : let n_1^{-1} be the multiplicative inverse of n_1 mod n_2, and n_2^{-1} be the multiplicative inverse of n_2 mod n_1

$f_i(j)$: let $f_i(j)$ be one kind in the private key

$f_i^k(j)$: let $f_i^k(j)$ be the kind in the public key corresponding to the kind $f_i(j)$ in the private key. k represents kth participant's public key, $k = (1, 2)$ in this small example.

private key:

Let f be the private key known to all k participants, which can be represented as follows:

$$f = \begin{pmatrix} (f_1(1), & f_1(2), & ..., & f_1(m)) \\ (f_2(1), & f_2(2), & ..., & f_2(m)) \\ (..., & ..., & ..., & ...) \\ (f_n(1), & f_n(2), & ..., & f_n(m)) \end{pmatrix}$$

public key:

$$f'^k = \begin{pmatrix} (f'^k_1(1), & f'^k_1(2), & ..., & f'^k_1(m)) \\ (f'^k_2(1), & f'^k_2(2), & ..., & f'^k_2(m)) \\ (..., & ..., & ..., & ...) \\ (f'^k_n(1), & f'^k_n(2), & ..., & f'^k_n(m)) \end{pmatrix}$$

Let f'^k be obtained by the operation "w times mod $n_k, (k = 1, 2)$ " on each kind in f. Each kind in f'^k can be expressed by $f'^k_i(j) = f_i(j) * w \,(mod\, n_k)$, $(k = (1, 2),\ i = (1, 2, ..., n),\ j = (1, 2, ..., m))$, and thus each kind in f can be expressed by $f_i(j) = f'^k_i(j) * w^{-1} \,(mod\, n_k), (k = (1, 2),\ i = (1, 2, ..., n),\ j = (1, 2, ..., m))$.

Ownerships of parameters/keys:

To be private	***To be public***
private key f a multiplier w the value of $N = \prod_{k=1}^{2} n_k$ each n_k and $N_k N_k^{-1}, (k = 1, 2), (N_k = \frac{N}{n_k},\ N_k^{-1} = (\frac{N}{n_k})^{-1})$	public key f'^k

Table 4.1: Ownerships of parameters/Keys

Some of the private parameters/keys shown in the above table are revealed to all participants when the system is set up, whereas others are only known to the combiner.

Specifically, each participant $P_k, (k = 1, 2)$ only knows the private key f, the multiplier w, and the product of N_k and N_k^{-1}, $(k = 1, 2)$. Each n_k is not known to each participant $P_k, (k = 1, 2)$. Only the trusted authority knows each $n_k, (k = 1, 2)$. Thus the trusted authority computes the kinds in the public key for each participant by $f'^k_i(j) = f_i(j) *$

$w\,(mod\,n_k)$, $(k = 1, 2)$, $(i = 1, 2, ..., n)$, $(j = 1, 2, ..., m)$, and distributes the value of $N_k N_k^{-1}$ to each participant securely. Then it disappears after the public keys are set up. As for the value of N, only an combiner knows its value.

shares generation:

1. Suppose the dealer wants to send an encrypted message M for all participants among the group, such that they can read it only when they all agree to recover it. Let message M be represented as the vector $\mathbf{x} = (x_1 x_2 ... x_n)$. The dealer computes two cryptograms C_1 and C_2 for those two participants by

$$C_1 = \sum_{i=1}^{n} f_i^1(x_i)$$

$$C_2 = \sum_{i=1}^{n} f_i^2(x_i)$$

2. The dealer distributes C_1 and C_2 to the corresponding participant P_k without worrying about being known by an attacker.

secret recovery:

1. When all participants decide to recover the secret, they compute the partial information of the secret by their own. The partial information is computed by $C_k * w^{-1} * (N_k N_k^{-1})$, $(k = 1, 2)$. Let N_k be $\frac{N}{n_k}$, and N_k^{-1} be $(\frac{N}{n_k})^{-1}$.

2. Each participant sends the partial result to the combiner

3. The combiner computes the value of the message M by

$$M = \sum_{k=1}^{2} C_k * w^{-1} * N_k N_k^{-1} (mod\, N)$$

4. Since $M = \sum_{i=1}^{n} f_i(x_i)$, the combiner recovers the secret by using the previous non-linear decryption scheme. Thus the secret is known by all participants at the same time.

example:

Let us take the following values for each parameter.

$w = 200$, $n_1 = 17$, $n_2 = 19$, $N = n_1 * n_2 = 323$, $w^{-1} = 21$, $n_1^{-1} = 9$, $n_2^{-1} = 9$, $n_1 n_1^{-1} = 153$, and $n_2 n_2^{-1} = 171$

Thus the private key is $f = \begin{pmatrix} (8, & 72, & 64) \\ (144, & 128, & 16) \\ (1, & 32, & 33) \\ (4, & 6, & 2) \end{pmatrix}$, the same as shown in the non-linear knapsack scheme. The public keys are different. They are computed as $f'^1 = \begin{pmatrix} (2, & 1, & 16) \\ (2, & 15, & 4) \\ (13, & 8, & 4) \\ (1, & 10, & 9) \end{pmatrix}$ and $f'^2 = \begin{pmatrix} (4, & 17, & 13) \\ (15, & 7, & 8) \\ (10, & 16, & 7) \\ (2, & 3, & 1) \end{pmatrix}$.

Let the message M be represented as the vector $\mathbf{x} = (1231)$, thus the ciphertexts are computed by $C_1 = 2 + 15 + 4 + 1 = 22$, and $C_2 = 4 + 7 + 7 + 2 = 20$. C_1 and C_2 are distributed to two participants.

When participants decide to recover the secret, they compute their own partial information, which are $C_1 * w^{-1} * N_k N_k^{-1} = 79002$ and $C_2 * w^{-1} * N_k N_k^{-1} = 64260$, respectively. They send the results to the combiner. The combiner adds two values of the partial information, and thus the value of the message is computed as $M = (79002 + 64260) \, (mod \, N) = 173$.

Applying the non-linear decryption scheme to the value of M, the combiner gets the secret message as the vector $\mathbf{x} = (1231)$, which is the same as the original value.

For generality, we let n_1, n_2, ..., n_k be numbers which are relatively prime, and let N be the product of all numbers $n_i, (i = 1, 2, ..., k)$. And then let $N_i = \frac{N}{n_i}$, and $M_i = N_i^{-1} \, (mod \, n_i)$ for $i = 1, 2, ..., k$. The public key is computed as

$$f'^l_i(j) = f_i(j) * w \, (mod \, n_l), \; for \, l = 1, 2, ...k$$

The dealer computes the cryptograms by $C_l = \sum_{i=1}^{n} f_i^l(x_i)$, $(l = 1, 2, ..., k)$, and distributes C_l, $(l = 1, 2, ..., k)$ to k participants.

Each participant computes their own partial information by $C_l * w^{-1} * N_l M_l$, $(l = 1, 2, ..., k)$. Then the results are sent to the combiner. The combiner computes the value of M by $M = \sum_{l=1}^{k} C_l * w^{-1} * N_l M_l \, (mod\, N)$, and then applies the non-linear knapsack scheme to recover the secret.

4.4 Analysis of Security

As we mentioned in the beginning of this chapter, Shamir's threshold secret sharing scheme is a perfect scheme, whereas Blakley scheme is not.

If a (t, n) secret sharing scheme is perfect, then even if any $t-1$ participants collude in order to recover the secret, they still would not determine more information about the secret than an outsider, who does not hold any shares. In Shamir's scheme, the collections of $t-1$ shares do not give any help to recover the secret. However, in Blakley's scheme, each participant with a share of the secret knows the secret is a point in his hyperplane. Therefore, the latter is not a perfect scheme, whereas the former is.

According to the above definition, our (n, n) threshold secret sharing scheme is not perfect. For simplicity of illustrations, we take $n = 2$ as an example. Suppose that b_1 and b_2 are two kinds in a participant P_1's public key, and a_1 and a_2 are two corresponding kinds in the private key. Thus b_1 and b_2 can be expressed by

$$b_1 = a_1 * w \, (mod\, n_1) = a_1 * w + k_1 n_1, \; (for\, some\, constant\, k_1)$$

$$b_2 = a_2 * w \, (mod\, n_1) = a_2 * w + k_2 n_1, \; (for\, some\, constant\, k_2)$$

where n_1 is a prime number. The above equations can be expressed as

$$a_2 b_1 - a_1 b_2 = (a_2 k_1 - a_1 k_2) n_1 \qquad (1)$$

If another pair of kinds (b_3, b_4) is found, then another equation (2) is made as follows:

$$a_4 b_3 - a_3 b_4 = (a_4 k_3 - a_3 k_4) n_1 \qquad (2)$$

Thus another participant within the group P_2 can obtain the value of n_1 by computing the value of $gcd(a_2b_1 - a_1b_2,\ a_4b_3 - a_3b_4)$ (all values of from a_1 to a_4 and from b_1 to b_4 are all known to P_2), which causes other secret values known to P_2, such as n_1^{-1}, and n_1n_2. That is to say, if P_2 somehow obtains the cryptogram C_1 from P_1, the secret would be revealed to P_2 without any help from another participant P_1.

In addition, in any perfect secret sharing schemes, if any participant had a share of bit-size less than that of the secret, the knowledge of the shares would reduce the security of the secret. Take the case of two participants for example. Suppose that the message M (denoted as an integer) has n bits, and $M < N$, where $N = n_1n_2$. By introducing two members, we reduce the size of N to half if the size of each n_k is nearly equal. Thus, the size of each share $C_k, (k = 1, 2)$ is $\frac{n}{2}$ bits. For generality, if the number of participants increases to k, the size of each participant's share is decreased to $\frac{n}{k}$ bits, if the size of secret is fixed. Thus for the authorized participants, we invite the exhaustive search easily because the uncertainty of the secret is reduced. However, if we fix the number of participants to k, and increase the size of the secret, the decryption time would increase. Therefore, for our scheme, we have to adjust those parameters depending on the situation.

Another weakness in our scheme is that it might not be convenient for the system to dynamically update when new users join in. In that case, the parameters $N_lM_l, (l = 1, 2, ..., k)$ have been updated and distributed over all participants. In addition, the participants' public keys have been updated as well. Therefore, it might be inconvenient for a large system to update.

Although our scheme has such several weaknesses, the advantage of our proposed scheme over other schemes would be that it increases the security against attacks from outsiders. That is to say, even if outsiders collect all the shares from the participants, they still could not recover the secret because each share is encrypted. As we can see from the description in section 4.3, when all participants decide to recover the secret, they compute the partial information by $C_l * w^{-1} * N_lM_l,\ (l = 1, 2, ..., k)$, and then they send their partial results to the combiner. For the combiner, he/she is the only person who knows the value of N. He/she computes the value of M, and then applies the non-linear decryption scheme to recover the secret. During this process, even if

an attacker collects all shares, it would be still hard for him/her to recover the secret, because for an attacker, the secret is encrypted by using non-linear knapsack scheme.

In contrast, for other schemes such as Shamir's (t, n) scheme t or more shares are pooled together, the combiner can easily recover the secret.

4.5 Conclusion

We have developed and presented in this chapter an encrypted (n, n) threshold secret sharing scheme. Although our scheme is not a perfect scheme, and dynamic updating is difficult, the merit of our system lies in the fact that we make the shares more secure by introducing the non-linear knapsack scheme.

Chapter 5

A Multiple Identities Authentication Scheme

5.1 Introduction

Firstly, let us imagine the following access control scenario. A given service S allows the users to access if they own privileges P_1, P_2, and P_3. That is to say, the users who own the superset of privileges P_1, P_2, and P_3 are allowed to gain access to this service S. This scenario actually involves a two-step process.

The first step is called *entity authentication*, also known as *identification*. *Entity authentication (identification)*, generally speaking, is a process by which a verifier gains assurance that the identity of a prover is as claimed, i.e., there is no impersonation [41]. Each identity is linked to a pair of private and public key. An identification scheme enables a prover holding a secret key to identify himself/herself to a verifier holding the corresponding public key.

The second step, *authorization*, is the process of granting or denying the users' access to the service based on the users' one or multiple identities. This two-step process is also viewed as users' needs to prove their one (or multiple) identities to the access control system.

A question of how to design such an access control system arises. One naive way is to link a different public key to each privilege. For example, if a user Alice owns the privileges P_1, P_2, and P_3, those three privileges will be correspondingly linked to three different public keys pk_1, pk_2, and pk_3. If she wants to gain access to the service S, she has to prove her ownership of the privileges P_1, P_2, and P_3 one by one to the access control system. That is to say, Alice as a prover, and Bob as a verifier (who actually plays the role of the access control system) have to perform any identification scheme d times, when d identities (Here identities could be users privileges which are linked to the user's public keys) are required for the access.

Another way is to use a public key certificate, which is a digital document signed by the Certification Authority (CA). It contains a specially formatted block of data, such as the certificate holder's name, the subset of privileges, and a public key, etc., which all serve to validate the holder's authorization. Thus, when a prover Alice and a verifier Bob perform an identification scheme based on Alice's public key certificate, Bob checks Alice's certificate to get the set of Alice's privileges. If Alice's privileges are the superset of the ones required for the access she is attempting, access is granted.

There is the third way, which is a more attractive method than the above two. It is called a Batching Schnorr's Identification Scheme, which was recently proposed by Gennaro et. al. [17]. His method greatly reduces the authentication time by doing a batch processing of Schnorr's identification scheme [55], and provides privacy-preserving property for the users. In addition, its computational complexity is suitable to be used in resource constrained devices.

We notice that a similar work, using the non-linear knapsack cryptosystem, can be done to provide the functionality of multiple identities authentication. More importantly, the computational complexity of our scheme is low as well. Our experimental results show high efficiency of our scheme.

Related work will be given in Section 5.2, which includes the overview of the first three ways to implement the access control system. Then our scheme will be discussed in detail in Section 5.3, including the description of our algorithm and proof. After that, the evaluation is presented in Section 5.4, which is focused on the analysis, and the

comparisons of efficiency and security, followed by possible applications of our scheme in Section 5.5. Finally we summarize the whole chapter in Section 5.6.

5.2 Related Work

5.2.1 d executions for d identities authentication

As we mentioned at the start of this chapter, a different public key is linked to each privilege. When Alice attempts to gain access to the service S, she has to prove the ownership of the privileges P_1, P_2, and P_3 to Bob, a verifier. Specifically speaking, in order to prove the ownership of the privilege P_1, Alice and Bob perform an identification scheme, using for example, the Schnorr's scheme. As shown in Figure 5.1, Alice initiates the Schnorr's scheme by sending her public key pk_1 to Bob (Alice public key pk_1 is linked to her privilege P_1.) Then Bob challenges Alice by sending a random number to Alice. If Alice generates the correct response based on value of the random challenge from Bob and knowledge of her own private key, and sends the response back to Bob, her ownership of the privilege P_1 will be accepted by Bob. Then Alice and Bob proceed to perform the Schnorr's scheme in order to prove Alice's ownership of the privilege P_2 which is linked with her public key pk_2, and so forth.

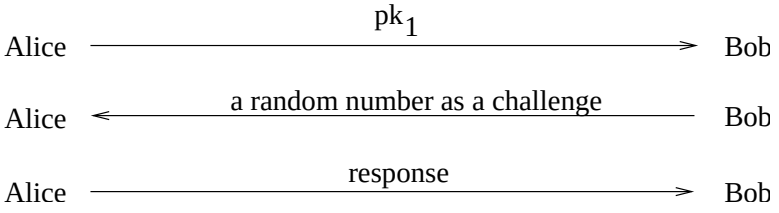

Figure 5.1: Verification of the Privilege P_1 by Using the Schnorr Scheme

The following illustrates the Schnorr's identification scheme in detail. Its security lies on the difficulty of discrete logarithm problem, which was discussed in Chapter 2. Once the private key w has been chosen one can easily compute the corresponding public key y, such that $y = g^{-w} \, (mod \, p)$ [55]. However, the inverse process, to compute w from

y, requires to compute the discrete logarithms with base g of y^{-1}, i.e., $w = -\log_g y$, which is assumed to be hard.

parameters:

p, q : let p and q be two prime numbers such that $q|p-1$

g : let g be an element not equal to 1, such that

$$g^q \equiv 1 \, (mod \, p)$$

w : let w be a random number less than q

y : let y be a number such that $y = g^{-w} \bmod p$

private keys:

w : a private key only known by the owner

public keys:

y : a public key published in a public place

ownership of parameters / keys:

To Be Private	***To Be Public***
private key w	public key y prime numbers p and q an element g

Table 5.1: Ownership of Parameters/Keys

actions of scheme:

First of all, Alice initiates a protocol by sending her public key pk_1 to Bob. Then Alice and Bob follow the process.

1. ***Commitment by a prover***.

   Firstly, Alice picks a random number r, such that r is less than $q-1$, and then computes x by the formula, $x = g^r \bmod p$. Alice sends x to Bob.

2. ***Challenge from a verifier***.

   Bob chooses a random number e such that $e \in [1...2^t]$, (t is a security parameter depending on the requirements of the application/system, which is pre-specified before the session starts) and sends it to Alice.

3. ***Response from a prover***.

   Alice computes the value of s by using the formula $s = r + w{*}e \pmod q$ and sends s as a response to Bob.

4. ***Verification by a verifier***.

   Bob checks that $x = g^s y^e \pmod p$ and accepts if and only if equality holds.

proof:

$x = g^s y^e \,(mod\, p)$

$= g^{(r+w*e+kq)} y^e \,(mod\, p)$ (as $s = r + w{*}e{+}kq$, for some constant k)

$= g^{(r+w*e+kq)} g^{(-w)*e} \,(mod\, p)$ (as $y = g^{-w} \,(mod\, p)$)

$= g^{(r+kq)} \,(mod\, p)$

$= g^r (g^q)^k \,(mod\, p)$

$= g^r \,(mod\, p)$ (as $g^q \equiv 1 \,(mod\, p)$)

$= x$

Obviously if Alice and Bob follow the protocol, then Bob will accept Alice's proof of identity [55]. The above process illustrates the verification of one identity. If Alice has d identities requested to be verified, the protocol has to be performed d times, as shown in Figure 5.2. Thus it is cumbersome and time-consuming.

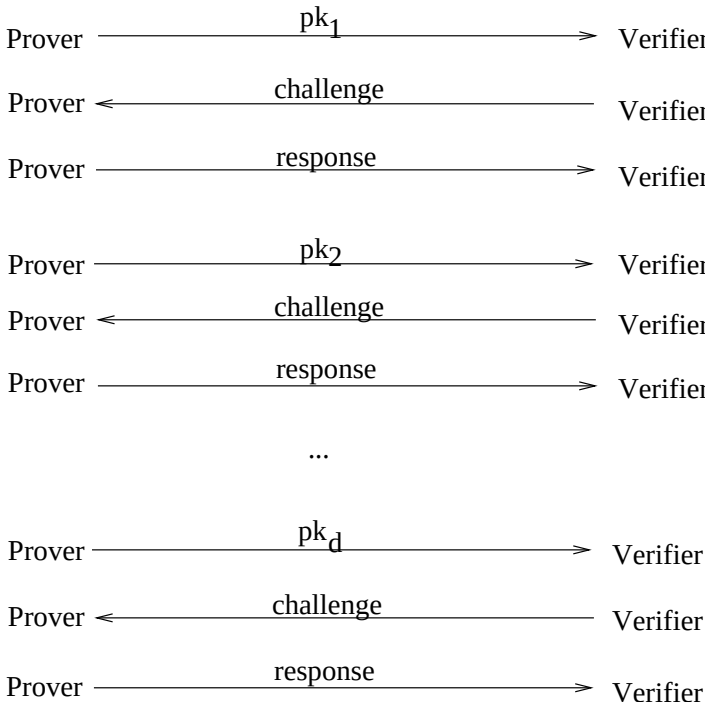

Figure 5.2: d Executions of an Identification Scheme for d Identities Authentication

5.2.2 A public-key certificate for multiple identities authentication

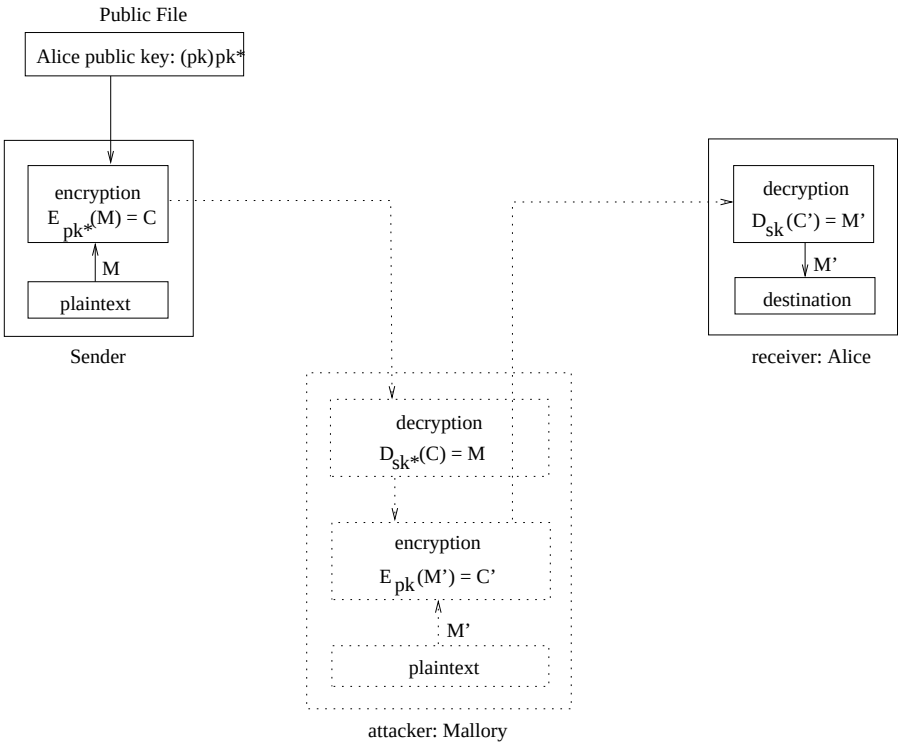

Figure 5.3: Impersonation by an Attacker with Public Key pk^*

As we can see from the above figure 5.3, when Alice publishes her public key pk and expects other persons to send the secret messages encrypted by her public key. A bad guy, Mallory, can also publish his public key $pk*$ (for which he knows the related private key) claiming it is Alice's. Thus, Mallory can decrypt secret messages M meant for Alice, if he intercepts the ciphertext C. Then Mallory may replace M with another message M', and sends the ciphertext C' to Alice. Therefore, Alice is cheated. In order to avoid this issue, it is obviously necessary that a principal's public key is associated with the principal's identity information in a verifiable and trustworthy way [38].

A public key certificate is a technique which uses a digital signature from the third trusted party (also known as the Certificate Authority, CA) to bind a public key with its holder, so that any other identities can verify that this public key really belongs

to that holder. Thus, it prevents the bad guy like Mallory from intentionally cheating the sender by claiming the ownership of the public keys.

A public key certificate is a formatted block of data with a number of data entries, including a uniquely identifiable identity of the holder and his/her public key parameter. The following table presents an example of X.509 public key certificate, which is one of the most common format of certificates.

Version
Serial Number
Algorithm Identifier: Algorithm and Parameters
Issuer
Period of Validity: Not before date, and Not after date
Subject
Subject's Public Key
Signature

Table 5.2: X.509 Certificate [22]

The overall process whereby Bob uses a public key certificate to obtain Alice's authentic public key may be summarized as follows [41]:

1. (One time) Acquire the authentic public key of the Certification Authority (CA).
2. Obtain an identifying string which uniquely identifies the intended party Alice.
3. Acquire over some unsecured channel a public key certificate corresponding to subject entity Alice and agreeing with the previous identifying string.
4. Verification

- Verify the current date and time against the validity period in the certificate, relying on a local trusted time

- Verify the current validity of the CA's public key itself
- Verify the signature on Alice's certificate, using the CA's public key
- Verify that the certificate has not been revoked.

If all checks succeed, then the public key in the certificate is accepted as Alice's authentic key.

With the widespread applications of a variety of Internet Services such as electronic businesses, or remote access and so on, a uniquely identifiable name with a key bound to it becomes inadequate [14]. In other words, the application needs to make a decision of whether the remote key holder is permitted to gain access to some services, when given a public key certificate. Therefore, a SPKI (Simple Public Key Infrastructure) certificate [14] was designed to meet this need.

A SPKI certificate is similar to an X.509 certificate. However, it consists of the authorization information, which means it can directly show to an application whether or not the request is authorized to perform an action [38]. In general, a SPKI certificate extends X.509 certificate to one with authorization features.

Compared with the first way mentioned in Section 5.2.1, d identities authentication using the SPKI certificate is only executed once. It is more efficient than the first way. However, the main weakness is the violation of the users' privacy. That is to say, whenever Alice proves her identity, she will reveal her all privileges in her certificate. It is unnecessary for her to reveal all of them, when she attempts to gain access to a given service. She should have revealed only a minimum subset of them. Thus, to some extend, Alice's privacy is violated.

5.2.3 A Batching Schnorr's scheme for multiple identities authentication

From the above discussions, an interesting question arises. Is there a way to verify d identities at the cost of less than d executions of an identification scheme and without

violating the user's privacy? Gennaro et. al. gives a positive answer to this question by generalizing the authentication scheme proposed by Schnorr [55], which is illustrated in Section 5.2.1.

His scheme [17], called a batching Schnorr's identification scheme enables the authentication of several identities at a cost very close to that of a single identity. In addition, it protects the privacy of the prover allowing him/her to prove only the minimum set of authorizations required to perform a given task, without disclosing that he/she is in possession of other privileges or not [17].

The detailed explanations of Gennaro's scheme are given as follows:

parameters:

p, q : let p and q be two prime numbers such that $q|p-1$

g : let g be an element such that $g^q \equiv 1\,(mod\,p)$, $1 \leq g \leq p-1$

d : the number of identities a user claims to own

w_i : let w_i be a random non-negative integer less than q, $1 \leq i \leq d$

y_i : let y_i be a number such that $y_i = g^{-w_i} \bmod p$, $1 \leq i \leq d$

private keys:

w_i : ith identity's private key that belongs to a user

public keys:

y_i : the corresponding ith identity's public key, such that $y_i = g^{-w_i} \bmod p$

ownership of parameters / keys:

To Be Private	*To Be Public*
private key w_i	public key y_i prime numbers p and q an element g

Table 5.3: Ownership of Parameters/Keys

actions of scheme:

A prover, Alice initiates the identification scheme by sending a list of her public keys for which it claims to possess the corresponding private keys [17]. The actions of scheme are described as follows:

1. ***Commitment by a prover***.

   A prover Alice, picks up a random number r, computes $x = g^r \bmod p$. Then Alice sends x to a verifier Bob.

2. ***Challenge from a verifier***.

   A verifier Bob picks a random number e as a challenge, such that $e \in [1...2^{(t+logd)}]$, and then sends it to Alice

3. ***Response from a prover***.

   Alice computes $s = r + \sum_{i=1}^{d} w_i * e^i (mod\, q)$ and sends s to Bob.

4. ***Verification by a verifier***.

   Bob checks that $x = g^s * \prod_{i=1}^{d} y_i^{e^i} (mod\, p)$, and accepts if and only if equality holds.

proof:

$$x = g^s * \prod_{i=1}^{d} y_i^{e^i} \,(mod\, p)$$

$$= g^{(r+\sum_{i=1}^{d} w_i * e^i + kq)} * \prod_{i=1}^{d} y_i^{e^i} \,(mod\, p) \quad (\text{as } s = r + \sum_{i=1}^{d} w_i * e^i + kq \text{ for some constant } k)$$

$$= g^r * g^{\sum_{i=1}^{d} w_i * e^i} * (g^q)^k * \prod_{i=1}^{d} (g^{-w_i})^{e^i} \,(mod\, p) \quad (\text{as } y_i = g^{-w_i} \bmod p)$$

$= g^r * (g^q)^k \, (mod\, p)$

$= g^r \, (mod\, p)$ (as $g^q \equiv 1 \, (mod\, p)$)

$= x$

As we can see from the above description that this scheme is a three-pass protocol. By replying with only one challenge from a verifier and performing the protocol only once, a prover can prove d identities. In addition, according to different requirements of authentication, a prover only needs to reveal the minimal subset of his/her privileges required to perform a given task. Therefore, this scheme provides a property of privacy-preserving. Its security, like the Schnorr's identification scheme, is based on the difficulty of discrete logarithm problem, described in section 2.4.2 in chapter 2.

5.3 Our Multiple Identities Authentication Scheme

Through our investigations into the non-linear knapsack cryptosystem, we notice that the non-linear knapsack system can be developed into an identification scheme similar to a batch Schnorr's identification scheme. More importantly, it is more efficient than the latter, because only simple multiple-precision arithmetic operations such as addition, multiplication, and arithmetic modular are involved.

We present our scheme with a small example in Section 5.3.1 in detail, followed by a proof in Section 5.3.2.

5.3.1 A multiple identities authentication scheme

parameters:**(We omit the description of some parameters appeared in the non-linear knapsack scheme.)**

d : number of identities a user claims to own

$f_i(j)$: the mapping from the ith item, and the jth kind to the $value(i,j)$ from the private key, ($i \in (1,...,n)$, $j \in (1,...,m)$), is denoted by $f_i(j)$. See the definition in section 3.5.1 for detailed description.

$f_i^{-1}(j)$: the mapping from the $value(i,j)$ to the ith item, and the jth kind is denoted by $f_i^{-1}(j)$

$f'^k_i(j)$: it is a value from the kth group's public key which the user owns, $k \in (1,...,d)$, $i \in (1,...,n)$, and $j \in (1,...,m)$

w_i : let w_i be a random number such that $\gcd(w_i, p) = 1$, $i \in (1,...,d)$

C_{ij} : let C_{ij} be some random integer numbers, which are $(i+1,j)$ elements of the verification matrix V

V : let V be a verification matrix, which is constructed in the following way:

$$V = \begin{bmatrix} w_1 & w_2 & ... & w_d \\ C_{11} & C_{12} & ... & C_{1d} \\ ... & ... & ... & ... \\ C_{(d-1)1} & C_{(d-1)2} & ... & C_{(d-1)d} \end{bmatrix}$$

V^{-1} : let V^{-1} be the inverse matrix of V mod p

R_i : let R_i be random integer number embedded in the messages which is less than a prime number p, $i \in (1,...,d-1)$

private keys:

$$f = \begin{pmatrix} (f_1(1), & f_1(2), & ..., & f_1(m)), \\ (f_2(1), & f_2(2), & ..., & f_2(m)), \\ ..., & ..., & ..., & ..., \\ (f_n(1), & f_n(2), & ..., & f_n(m)) \end{pmatrix}$$

public keys:

$$f'^k = \begin{pmatrix} (f'^k_1(1), & f'^k_1(2), & ..., & f'^k_1(m)), \\ (f'^k_2(1), & f'^k_2(2), & ..., & f'^k_2(m)), \\ ..., & ..., & ..., & ..., \\ (f'^k_n(1), & f'^k_n(2), & ..., & f'^k_n(m)) \end{pmatrix},$$

such that $f'^k_i(j) = f_i(j) * w_k(\text{mod p}),\ k \in (1, ..., d)$

ownership of parameters/keys:

To Be Private	***To Be Public***
private key f	
the multiplier w_i	public key f'^i
the prime number p	the matrix V of the rank $d-1$, (excluding the first row)
the matrix V,V^{-1}	

Table 5.4: Ownership of Parameters/Keys

actions of scheme:

As the batching Schnorr's scheme mentioned above, a prover Alice initiates the scheme by sending a list of her public keys for which it claims to possess the corresponding private keys. Then a verifier Bob generates a challenge, which is a ciphertext generated by using Alice's public keys to encrypt a random message, say X. If Alice replies with the same message as the original one X generated by Bob, Bob will accept what Alice claims. Otherwise Bob refuses it.

In general, our scheme is more like an encryption/decryption scheme. That is to say, Bob encrypts a random message by using Alice's public keys, and sends the ciphertext to Alice. If Alice is an honest claimant, which means she holds the corresponding private keys, she can decrypt the ciphertext and send the authentic message back to Bob. Then Bob will accept the identities that Alice claims to hold. It is difficult for Alice to compute the original message from the ciphertext, if she wants to cheat Bob.

The actions of this scheme are described as follows:

1. ***Challenge from a verifier.***

   Let $\mathbf{x}$ be a vertical vector $(X, R_1, R_2, ..., R_{d-1})^T$, and a message X be an m-ary sequence of length n. Let X be embedded into $\mathbf{x}$**.** Bob generates $d-1$ random number R_k which are all less than p. And Bob computes d ciphertexts C_k by

$$C_k = \sum_{i=1}^{n} f'^{k}_{i}(x_i) + \sum_{i=1}^{d-1} C_{ik} R_k$$

   Then Bob sends d ciphertexts C_1, ..., C_d to the prover Alice, who claims to own d identities.

   For example, we go with $d = 3$ for illustration. Thus,

$$C_1 = \sum_{i=1}^{n} f'^{1}_{i}(x_i) + R_1 + R_2$$

$$C_2 = \sum_{i=1}^{n} f'^{2}_{i}(x_i) + R_1 + 2R_2$$

$$C_3 = \sum_{i=1}^{n} f'^{3}_{i}(x_i) + R_1 + 3R_2$$

   In matrix form, we could have:

$$(C_1, C_2, C_3) = (\sum_{i=1}^{n} f_i(x_i), R_1, R_2) * \begin{bmatrix} w_1 & w_2 & w_3 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \qquad (5.1)$$

   For generality, we could have the following equation:

$$(C_1, ..., C_d) = (\sum_{i=1}^{n} f_i(x_i), R_1, ..., R_{d-1}) * V$$

2. ***Response from a prover***.

After Alice receives d ciphertexts, she starts to compute the value of $\sum_{i=1}^{n} f_i(x_i)$ in the following way:

$$(\sum_{i=1}^{n} f_i(x_i), R_1, ..., R_{d-1}) = (C_1, ..., C_d) * V^{-1} (mod\, p)$$

For example, as the following equation (5.2) shows, Alice computes $\sum_{i=1}^{n} f_i(x_i)$ by $(\sum_{i=1}^{n} f_i(x_i), R_1, R_2)$

$$= (C_1, C_2, C_3) * \begin{bmatrix} w_1 & w_2 & w_3 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}^{-1} (mod\, p)$$

$$= (C_1, C_2, C_3) * \begin{bmatrix} 1 & 2w_3 - 3w_2 & w_2 - w_3 \\ -2 & 3w_1 - w_3 & w_3 - w_1 \\ 1 & w_2 - 2w_1 & w_1 - w_2 \end{bmatrix} * (w_1 - 2w_2 + w_3)^{-1} (mod\, p)$$

It is obvious that Alice only needs to compute the inner product of $(C_1, ..., C_d)$ and first column of the inverse matrix $V^{-1}(mod\, p)$, which takes $O(d)$ operations of multiple-precision numbers.

Then Alice applies the original non-linear knapsack decryption method (which is given in the previous section) to $\sum_{i=1}^{n} f_i(x_i)$. Thus, Alice can recover the original message X, and send it to Bob.

3. ***Verification from a verifier***.

Bob checks the received message X with his original message. If two values are identical, Bob accepts d identities that Alice claims. Otherwise Bob refuses them.

5.3.2 Proof

Generally speaking, the processes of encrypting and decrypting rely on the properties of matrix reversibility. That is to say, if we multiply a horizontal vector $\mathbf{x}$ with a regular (d, d) matrix V to compute $c = \mathbf{x}V$, where $c = (C_1, C_2, ..., C_d)$, we can distribute the secrecy of X into n pieces of information c. By putting them together, and computing $\mathbf{x} = cV^{-1}$, we can read X. If we generate the number w_i and constant number C_{ij} randomly, the probability that the matrix V is singular is low.

Firstly, let us go with a small example ($d = 3$), and $V = \begin{bmatrix} w_1 & w_2 & w_3 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}$ for illustration.

As we can see from the equation (5.1),

$$(C_1, C_2, C_3) = (\sum_{i=1}^{n} f_i(x_i), R_1, R_2) * \begin{bmatrix} w_1 & w_2 & w_3 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

Thus the following equation holds (from the equation (5.2)):

$$(\sum_{i=1}^{n} f_i(x_i), R_1, R_2)$$
$$= (C_1, C_2, C_3) * \begin{bmatrix} 1 & 2w_3 - 3w_2 & w_2 - w_3 \\ -2 & 3w_1 - w_3 & w_3 - w_1 \\ 1 & w_2 - 2w_1 & w_1 - w_2 \end{bmatrix} * (w_1 - 2w_2 + w_3)^{-1} \, (mod\, p)$$

As we can see from the above equations, if the prover Alice knows the inverse matrix V^{-1}, she could compute the value of $\sum_{i=1}^{n} f_i(x_i)$ and then recover the message. In fact, Alice obtains (C_1, C_2, C_3) from the verifier Bob, and she knows the inverse matrix V^{-1}. It would be no problem for her computer $(\sum_{i=1}^{n} f_i(x_i), R_1, R_2)$, although she only needs to computer the value $\sum_{i=1}^{n} f_i(x_i)$. Then Alice applies the non-linear knapsack decryption process to that value in order to recover the message $\mathbf{x}$.

For generality, a secret message X is hidden in d pieces of information. The d ciphertexts C_1,..., C_d is computed by the following equation:

$$(C_1, C_2, ..., C_d) = (\sum_{i=1}^{n} f_i(x_i), R_1, ..., R_{d-1}) \begin{bmatrix} w_1 & w_2 & ... & w_d \\ C_{11} & C_{12} & ... & C_{1d} \\ ... & ... & ... & ... \\ C_{(d-1)1} & C_{(d-1)2} & ... & C_{(d-1)d} \end{bmatrix},$$

If the matrix V is regular, that is, invertible, we can have the following equation,

$$(\sum_{i=1}^{n} f_i(x_i), R_1, ...R_{d-1}) = (C_1, C_2, ..., C_d) * \begin{bmatrix} w_1 & w_2 & ... & w_d \\ C_{11} & C_{12} & ... & C_{1d} \\ ... & ... & ... & ... \\ C_{(d-1)1} & C_{(d-1)2} & ... & C_{(d-1)d} \end{bmatrix}^{-1} (mod\, p)$$

The prover, Alice, only needs to compute the inner product of $(C_1, ..., C_d)$ and first column of the inverse matrix of $V^{-1}(mod\, p)$, which takes $O(d)$ operations of multiple-precision numbers. Then applying the non-linear decryption scheme, the vector $\mathbf{x}$ can be decrypted from the value $\sum_{i=1}^{n} f_i(x_i)$. In addition, only the owner of the private key can compute the value of $\sum_{i=1}^{n} f_i(x_i)$, and recover the vector $\mathbf{x}$.

5.4 Evaluations and Comparisons

5.4.1 Efficiency analysis

(1) Time complexity

Firstly, let us analyze the time complexity of d executions of the Schnorr's identification scheme for d identities authentication. The prover requires to perform d multiplications. The verifier has to perform $2d$ modular exponentials.

Then, let us take a look at the time complexity of the batching Schnorr's scheme. For a list of d identities, the prover requires $2d$ modular multiplications. The verifier has to perform $(d + 1)$ modular exponentiations.

However, in our scheme, the verifier performs d multiplications, and $d*(d-1)$ additions, which is dominant. As for the prover, only d multiplication are required.

The following table shows the comparisons of time complexity among these three schemes. We use the standard $O(n^2)$ algorithm for multiplication and division of n- bit integers.

	d executions of the Schnorr scheme	Batch-Schnorr scheme	our scheme
Prover	$O(dn^2)$	$O(2dn^2)$	$O(dn^2)$
Verifier	$O(2dn^3)$	$O((d+1)n^3)$	$O(d^2n)$

Table 5.5: Comparisons of Time Complexity

(2) Experiments on running time

The following part shows the comparisons of experimental results on running time between our scheme and the batch Schnorr scheme.

As Gennaro et. al. mentioned in their paper, their scheme was implemented using the Microchip PICI6LF628 micro-controller. The PIC uses 8-bit instructions words and runs at 5 MIPS (million instructions per second), and it has 16KB of write-able storage [17]. The selections of security parameters are as follows:

Number of identities d: $d = 32$

Prime number q: let q be 200 bits

Prime number p : let p be about 1500 bits

Gennaro et. al. enabled the implementation to run in less than 2 seconds for their choice of security parameters.

As for the implementation of our scheme, we program in the C language on the Linux machine with a Celeron processor with 2.2 GHz. In order to treat two schemes under the same condition, our selections of parameters are as follows:

Number of identities d: $d = 32$

Number of items $n : n = 75$

Number of kinds $m : m = 10$

Number of mask bits for each item $l : l = 20$

Each mask pattern's length $ln : ln = 20 * 75 = 1500$

Prime number $p : p > 2^{ln}$. So $p > 2^{1500}$, p is about 451 decimal digits, which has a comparable size with the Batch-Schnorr scheme.

We encrypt a file with 24 characters as a challenge, and expect the intended receiver to decrypt it. The length of plaintext depends on the number of items, which makes the number of m-ary vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ be one. The following table shows our experimental results.

Experiments)	E(Encryption) (seconds)	D(Decryption) (seconds)	E+D (seconds)
1st Experiment	0.028	0.029	0.057
2nd Experiment	0.027	0.030	0.057
3rd Experiment	0.027	0.029	0.056
Average	***0.027***	***0.029***	***0.057***

Table 5.6: Running Time of Our Scheme

Compared with the running time of Batch-Schnorr scheme, ours only takes about 0.06 second, which greatly reduces the time. However, we also notice that our scheme runs in a much faster chip. Thus for the future work we need to find out some ways to further reduce the running time of our scheme, for example integrating the look-up table into our implementation.

(3) Memory requirements

As for the memory requirement for our selections of security parameters, we have the following calculation:

Memory requirement = $l*m*n^2$= $20*10*75^2$ ***bits*** = 1125000/8 ***bytes*** = 140625/1024 ***KB*** = 137 ***KB***

From the time complexity point of view, our scheme takes less time than the Batch-Schnorr scheme. However, the memory requirement of ours is about 10 times more than the latter. This disadvantage will not be a great obstacle to our system, because the cost of memory chip is going down frequently.

5.4.2 Security analysis

Generally speaking, we integrate the property of matrix reversibility into a non-linear knapsack cryptosystem to invent a multiple identities authentication scheme. In other words, the security of this scheme lies on the non-linear knapsack scheme, which is based on an NP-complete problem. As we mentioned before, for those problems, so far we have not been able to come up with any algorithm substantially better than searching through all possible designs [16]. Therefore, we can assume that our scheme is secure enough to resist any impersonation attack.

For impersonators, if they can give the proper response to the verifier, their claims will be accepted. However, it would not be easy for them, because recovering the vector $\mathbf{x}$ from the value of $\sum_{i=1}^{n} f_i(x_i)$ is NP-complete, which means they have to solve this problem in nondeterministically polynomial time. Such a class of problems has been studied by mathematicians and computer scientists for decades, but polynomial time bounded algorithms have not been found for even one of them. Therefore, we regard such problems as being intractable.

As for those known attacks to the linear knapsack cryptosystems, as we mentioned before, we have not found that they apply to our scheme. Firstly, the super-increasing sequence is not used to construct the private key, which is the main problem that causes the linear knapsack scheme to be broken. Furthermore, the generation of each item for the private key is non-linear. Thus linear attacks do not apply to our scheme.

On the other hand, as we suggested in chapter 3, we scatter l/2 bits for 1 and the rest for 0 to each kind in the private key. In addition, the number of kinds m could not be

too large to avoid the equal-sum event. That is to say, if an attacker could find another set of public keys kinds satisfying the equal-sum event, they could compute the value of p by using the Euclidean algorithm, which causes the break of the system. For security , we scatter l/2 ones for each mask, and then exhaustively check the private keys to avoid the equal-sum event. The detailed discussions can be found in section 3.6.1 in chapter 3.

5.4.3 Possible applications of our scheme

As we mentioned in the beginning of this Chapter, our scheme is suitable to be used in access control. That is to say, when there are several identities need to be verified to decide the authorization, our scheme could be a good choice. Moreover, it could be also secure without unnecessarily revealing any knowledge of the identity's privileges.

One application of our scheme is in production processing system, where a product needs to go through several processing steps, depending on the accesses granted.

Another interesting application is in the RFID (Radio Frequency IDentification) tags for access control. Our scheme satisfies the features of processing speed, privacy and authentication. In addition, the memory requirements is also in the reasonable level.

5.5 Conclusion

In this chapter, we introduce our second scheme, called a multiple identities authentication scheme. This scheme is much more efficient than other schemes with the similar functionalities. Moreover, our scheme provides the property of privacy-preserving.

We firstly reviewed other similar schemes, followed by the description of our scheme. Then we compared those schemes from two aspects, namely efficiency analysis and security analysis. Finally, we talked about the possible applications of our scheme to conclude this chapter.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

As a typical cryptosystem based on the knapsack problem, which is an NP-complete problem, the Merkle-Hellman cryptosystem has an obvious advantage over other cryptosystems which are based on different mathematical problems. It has the high efficiency. Compared with the widely used RSA cryptosystem, it can be more than 100 times faster when the modulus of the RSA system is about 500 bits.

However, the Merkle-Hellman cryptosystem and its many variants were broken, and it is not regarded as important any more in the cryptographic community. In order to revive the area of cryptosystems based on an NP-complete problem, Kiriyama recently proposed a new public key cryptosystem called the non-linear knapsack cryptosystem. Its security is based on the NP-completeness of the non-linear knapsack problem. We believe that this new system brings the theoretical breakthroughs to the research of knapsack cryptosystems, and the extension of his work is our main research objective.

Cryptography has a long and fascinating history, which can be traced back thousands of years. Chapter 2 introduces the main contributions to the development of cryptography, which starts with the earliest forms of ciphers, namely transposition, substitution ciphers (Caesar cipher) and one-time pad. Then we overview two types of key-based

cryptographic algorithms, which are symmetric cryptographic algorithms and asymmetric cryptographic algorithms. We introduce DES (Data Encryption Standard) and AES (Advanced Encryption Standard) as the typical symmetric cryptosystems. Then we present two categories of asymmetric cryptosystems. The first category is based on the intractability of integer factorization and include the Rabin cryptosystem and the RSA cryptosystem. The second one is based on the discrete logarithm problem. The ElGamal cryptosystem is a typical system in this category.

The third category of cryptosystem is based on an NP-complete problem. This presents in chapter 3. Firstly, we start with a general overview of the theory of NP-completeness, followed by an introduction of the Merkle-Hellman cryptosystem. As we mentioned before, the Merkle-Hellman cryptosystem and its variants were all broken. We generally overview Shamir's method for attacking to the Merkle-Hellman knapsack cryptosystem. After that we present the main foundation of our research, the non-linear knapsack cryptosystem. The evaluation of this scheme is given.

In chapter 4, we start to describe our work. The first scheme we invent is called an encrypted (n, n) threshold secret sharing scheme. Only when n members in a group all agree to pool their respective shares together, the plaintext can be recovered. We combine the concept of the CRT (Chinese Remaindering Theorem) and the non-linear knapsack scheme with the concept of secret sharing. Compared with other secret sharing schemes, the increased security and high efficiency are the main merits of our scheme.

Chapter 5 describes our second scheme, which is called a multiple identities authentication scheme, developed on the basis of the non-linear knapsack scheme. Simply speaking, it verifies the ownership of an entity's several identities in one execution of the scheme. Moreover, due to low computational complexity of our scheme, it can be used in resource-constrained devices for access control.

Each of our schemes is implemented in the C language under the Linux system, and we evaluate the running time for each. The experimental results show that our schemes are efficient due to low computational complexity of the non-linear knapsack problem.

6.2 Future Work

By combining the non-linear knapsack scheme with other mathematical concepts, we proposed several new schemes. That is to say, all of our research work is based on the non-linear knapsack scheme. In addition, the security of our schemes rests on the unbreakability of the non-linear knapsack scheme. As we discussed before, the non-linear knapsack scheme is based on the non-linear knapsack problem, which is analogous to the knapsack problem, the basis of the Merkle-Hellman knapsack scheme. The knapsack problem has already been proven as an NP-complete problem. However, we are unable to prove that the difficulty of breaking the non-linear knapsack cryptosystem is equivalent to that of solving a non-linear knapsack problem. If anyone could prove it, that would be an exciting point. Moreover, the non-linear knapsack cryptosystem would have a strong mathematical foundation.

We have investigated existing attack methods to the Merkle-Hellman knapsack scheme (for example Shamir's attack method). We have found they could not directly be used to attack the non-linear knapsack scheme. We welcome any efforts that would try to break the non-linear knapsack scheme.

As for the (n, n) threshold secret sharing scheme presented in chapter 4, initially we expect we could achieve privacy within a group. That is to say, even though a group member illegally obtains all n shares from others, he/she still could not recover the secret. Unfortunately, this goal was not achieved. This is an interesting further research topic.

Bibliography

[1] Alexi W., Chor B.-Z., Goldreich O., and Schnorr C. P., *RSA and Rabin Functions: Certain Parts are as hard as the Whole*, SIAM Journal on Computing, v.17, n.2, Apri. 1988, pp.194-209

[2] Asmuth C. and Bloom J., *A Modular Approach to Key Safeguarding*, IEEE Transactions on Information Theory IT-29, 1983, pp.208-211

[3] Atkin A. O. L. and Larson R. G., *On a Primality Test of Solovay and Strassen*, SIAM Journal on Computing, 11 (1982), 789-791

[4] Blakley G. R., *Safeguarding Cryptographic Keys*, Proc. AFIPS 1979 Natl. Computer Conf., New York, vol. 48, pp.313-317, June 1979

[5] Boneh Dan, *Twenty Years of Attacks on the RSA Cryptosystem*, Notices of the AMS, 46(2):203–213, 1999

[6] Branstad D. K., Gait J., and Katzke S., *Report on the Workshop on Cryptography in Support of Computer Security*, NBSIR 77-1291, National Bureau of Standards, Sep 21-22, 1976, September 1977

[7] Brickell E. F., *Some Ideal Secret Sharing Schemes*, Journal of Combinatorial Mathematics and Combinatorial Computing 6, 1989, pp. 105-113

[8] Chor Benny, *A knapsack Type Public Key Cryptosystem Based on Arithmetic in Finite Fields*, IEEE Transaction on Information Theory, Vol 34, 1988, pp. 901-909

[9] Cormen Thomas H., Leiserson Charles E., Rivest Ronald L., and Stein Clifford, *Introduction to Algorithms*, Second Edition, the MIT Press, Cambridge, Massachusetts, London, england, 2002

[10] Daemen Joan and Rijmen Vincent, *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer-Verlag, 2002

[11] Desmedt Y., Vandewalle J., and Govaerts R., *Critical Analysis of the Security of Knapsack Public Key Algorithms*, Proc. 3rd Symposium on Information Theory in the Benelux, Zoetermeer (NL), May 13-14, 1982, pp. 19-27

[12] Diffie W. and Hellman M. E., *New Direction in Cryptography*, IEEE Trans. Informat. Theory., vol. IT-22, pp. 644-654, Nov. 1976

[13] ElGamal T., *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, Advances in Cryptology, Proceedings of CRYPTO 84, Springer - Verlag, 1985, pp. 10-18

[14] Ellison C., Frantz B., Lampson B., Rivest R., Thomas B., and Ylonen T., *SPKI Certificate Theory, The Internet Task Force Request for Comments (IETF RFC) 2693*, September 1999, available at SPKI, `http://www.ietf.org/rfc/rfc2693.txt`

[15] Gaines H. F., *Cryptanalysis*, American Photographic Press, 1937 (Reprint by Dover Publications, 1956.)

[16] Garey Michael R. and Johnson David S., *Computers and Intractability: a Guide to the Theory of NP-Completeness*, San Francisco: Freeman, (1979)

[17] Gennaro R., Leigh D., Sundaram R., and Yerazunis W., *Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authentication and Low-Bandwidth Communication Devices*, AsiaCrypt 2004, LNCS 3329, pp. 276-292, Springer-Verlag Berlin Heidelberg 2004

[18] Gilmore John, *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*, 1998, published by O'Reilly and Association

[19] Gustavus J.Simmons, *A Weak Privacy Protocol Using the RSA Crypptographic Algorithm*,Cryptology, 7(2), April. 1983

[20] Hopcroft John E. and Ullman Jeffrey D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979

[21] IEEE. *P1363: Standard Specifications for Public-Key Cryptography*, available at `http://www.manta.ieee.org/groups/1363/P1363/index.html`

[22] ITU-T, Rec. *X.509 (revised) the Directory - Authentication Framework*, 1993. International Telecommunication Union, Geneva, Switzerland (equivalent to ISO/IEC 9594 - 8: 1995)

[23] Kahn D., *the Codebreakers: The Story of Secret Writing*, New York: Macmillan Publishing Co., 1967

[24] Karin E. D., Greene J. W. and Hellman M. E., *On Secret Sharing Systems*, IEEE Transactions on Information Theory IT-29, 1983, pp. 35-41

[25] Kiriyama Akito, *An Application of Discrete Optimization to Cryptology*, Master Thesis, School of Information, Kansai University, March 2005

[26] Kiriyama Akito, Nakagawa Yuji, Takaoka Tadao and Tu Zhiqi,*A New Public-Key Cryptosystem and its Application*, to appear in Proceedings of 8th International Conference on Enterprise Inforamtion Systems, 23 - 27, May 2006 (ICEIS 2006)

[27] Knuth D. E., *The Art of Computer Programming*, Vol 2: Seminumerical Algorithms. Addison-Wesley, Reading, Mass., 1969

[28] Knuth D. E., *The Art of Computer Programming*, Vol. 2, Seminumerical Algorithms, Addison-Wesley, 1981 (Second Edition)

[29] Knuth D. E., *Seminumerical Algorithms*, volume 2 of The Art of Computer Programming, Addison-Wesley, Reading, massachusetts, Second Edition, 1981

[30] Kubale Marek, *Introduction to Computational Complexity*, Gdansk [Poland]: Politechnika Gdanska, 1994

[31] Kulesza Kanil, and Kotulski Zbigniew, *On Secret Sharing Schemes with Extended Capabilities*, Proceedings of Regional Conference on Military Communication and Information Systems, CIS Challenges in the New Environment, RCMIS 2002, October 9th-11th 2002, Zegrze, vol. 1. pp.79-88

[32] Lagarias J. C., and Odlyzko A. M., *Solving Low Density Subset Sum Problems*, JACM, vol. 32, 229-246, 1985

[33] Lai Ming Kin, *Knapsack Cryptosystems: The Past and the Future*, March 2001, available at `http://www.cecs.uci.edu/ming/knapsack.html`

[34] LaMacchia B. A. and Odlyzko A. M., *Computation of Discrete Logarithms in Prime Fields*, Designs, Codes, and Cryptography, v. 1, 1991, pp. 46-62

[35] Lenstra A. K., Lenstra H. W. and Lov'asz L., *Factoring polynomials with Rational Coefficients*, Mathematische Annalen 261, 515-534, 1982

[36] Lenstra H. W., Jr., *Integer Programming with a Fixed Number of Variables*, Math, Operations Research, vol. 8, 1983, pp. 538-548

[37] Lewis Harry R. and Papadimitriou Christos H., *Elements of the Theory of Computation*, Prentice-Hall, second edition, 1998

[38] Mao Wenbo, *Modern Cryptography: Theory and Practice*, Prentice Hall PTR, 2004

[39] Massey James L., *Contemporary Cryptology: An Introduction*, in Comtemporary Cryptology: The Science of Information Integrity, G.J. Simmons, ed., IEEE Press, 1992, pp. 1-39

[40] Menezes Alfred J., Blake I. F., Gao X., Mullin R. C., Vanstone S. A., and Yaghoobian T., *Applications of Finite Field*, Kluwer, Boston, 1993

[41] Menezes Alfred J., Paul C. van Oorschot, and Vanstone Scott A., *Handbook of Applied Cryptography*, CRC 1996

[42] Merkle L. C., and Hellman M. E., *Hiding Information and Signatures in Trapdoor Knapsacks*, IEEE Trans. On Information Theory, 24, pp. 525-530, 1978

[43] Mitchell C. J., Piper F., and Wild P., *Digital Signatures*, in Contemporary Cryptology: The Science of Information Integrity, G. J. Simmons, ed., IEEE Press, 1991, pp. 325-378

[44] National Institute of Standards and Technology, *NIST FIPS PUB 186, Digital Signature Standard*, U. S. Department of Commerce, May 1994

[45] National Bureau of Standards, *Data Encryption Standard*, FIPS-Pub. 46. National Bureau of Standards, U. S. Department of Commerce, Washington D. C., January 1997

[46] Odlyzko A. M., *Discrete Logarithm in Finite Field and Their Cryptographic Significance*, Advances in Cryptology - EUROCRYPT'84, Lecture Notes in Computer Science, Volume 209, Springer-Verlag, Berlin, 1985, 224-314

[47] Odlyzko A. M., *the Rise and Fall of Knapsack Cryptosystems*, Cryptology and Computational Number Theory, Am. Math. Soc., Proc. Symp. Appl. Math., Vol. 42, pp. 75-88, 1990

[48] Odlyzko A. M., *Discrete Logarithms and Smooth Polynomials, in Finite Fields: Theory, Applications, and Algorithms*, G. L. Mullen and P. J. S. Shiue (eds.), Contemporary Mathematics, Volume 168, American Mathematical Society, Providence, RI, 1994, 269-278

[49] Odlyzko A. M., *Discrete Logarithms: the Past and the Future*, Designs, Codes, and Cryptography 19 (2000), pp. 129-145. Reprint in Towards a Quarter-Century of Public Key Cryptography, N. Koblitz, ed., Kluwer, 2000, pp. 59-75

[50] Pieprzyk Josef, Hardjono Thomas, and Seberry Jennifer, *Fundamentals of Computer Security*, Springer-Verlag Berlin, Heidelberg, New York, 2003

[51] Rabin M. O., *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, MIT/LCS/TR-212, MIT Lab. for Computer Science, 1979

[52] Rivest R. L., Shamir A., and Adelman A., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystem*, CACM, 21, 2, pp. 120-126, 1978

[53] RSA Laboratories, available at `http://www.rsasecurity.com/rsalabs`

[54] Schneier Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Second Edition, CRC 1996

[55] Schnorr C. P., *Efficient Signature Gneration for Smart Cards*, Journal of Cryptology, vol. 4, no. 3, pp. 161-174 (1991)

[56] Shamir A., *How to Share a Secret*, Massachusetts Institute of Technology Technical Report MIT/LCS/TM-134, May 1979

[57] Shamir A., *Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem*, FOCS 1982: 145-152

[58] Shannon C. E., *Communication Theory of Secrecy Systems*, Bell Syst. Tech. J., vol. 28, pp. 656-715, Oct. 1949

[59] Sinkov A., *Elementary Cryptanalysis*, Mathematical Association of Amercia, 1996

[60] Solovay R. and Strassen V., *A Fast Monte-Carlo Test for Primality*, SIAM Journal on Computing, 6 (1977), 84-85, Erratum in ibid, 7 (1978), 118

[61] Studholme Chris, *The Discrete Log Problem*, Research Paper, June 21, 2002, available at `http://www.cs.toronto.edu/cvs/dlog/research\_paper.pdf`

[62] Takaoka T and Williams R, *A Better Method of Obtaining Signature in the Knapsack Cryptosystem*, Lecture Notes in Department of Computer Science, University of Canterbury, Christchurch, New Zealand

[63] Wiki, available at `http://en.wikipedia.org/wiki/Cryptanalysis`, version 02:37,14 March 2006

[64] Wu C.K. and Wang X.M., *Determination of the True Value of the Euler Totient function in the RSA Cryptosystem from a Set of Possibilities*, Electronics Letters, v.29, n.1, 7 Jan. 1993, pp.84-85

[65] Zhou Lidong, Lecture Notes for Secret Sharing, available at `http://www.cs.cornell.edu/Courses/cs513/2000SP/SecretSharing.html`